UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

BGC :
BIBLIOTHÈQUE DE FONCTIONS GRAPHIQUES EN LANGAGE C

Première version

par

Jean-François Mondou
Teodor Gabriel Crainic
Sylvie Couture

Cette recherche a été rendue possible grâce à une subvention du C.R.S.N.G. et de F.C.A.R.

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour l'aide et le soutien à la recherche.

Université de Montréal

Centre de recherche sur les transports - Publication #554
Département d'informatique et
de recherche opérationnelle - Publication #641

Décembre 1987

Dépôt légal - 1er trimestre 1988

RÉSUMÉ

Nous présentons une nouvelle bibliothèque graphique écrite en langage C. Les récents développements des périphériques graphiques à des prix abordables ont permis la diffusion de nombreux systèmes graphiques dont les applications sont très variées. Les fonctions de notre bibliothèque graphique sont bien adaptées au graphisme de réseau et nous croyons qu'elles constituent un noyau solide pour la mise au point de systèmes interactifs graphiques pour les problèmes de transport et de localisation de dépôt. Ce rapport est divisé en trois sections. Nous commencerons par énoncer brièvement les concepts de base qui sous-tendent la mise en oeuvre d'une bibliothèque graphique. Ensuite nous décrirons la structure de BGC et nous terminerons par une description exhaustive des fonctions qui la composent.

Mots-clés : bibliothèque graphique; langage C; graphisme de réseau; système interactif graphique.

INTRODUCTION

Notre mécanisme oeil-cerveau de reconnaissance de formes nous permet d'analyser rapidement plusieurs types de données lorsqu'elles sont présentées graphiquement.

L'informatique graphique est la création, l'emmagasinement et la manipulation de modèles d'objets ainsi que leur représentation graphique à l'aide d'un ordinateur. Un cas particulier est l'informatique interactive graphique où l'usager contrôle dynamiquement les images générées. Un environnement de travail est créé et une conversation est établie entre l'ordinateur et l'utilisateur via des menus, icônes et dispositifs d'entrée tel le clavier, la souris, la tablette graphique et autres.

L'apparition des périphériques graphiques performants (moniteur couleur, imprimante à points et au laser, etc.) à des prix abordables ont permis la diffusion des systèmes graphiques. Il n'y a qu'à penser aux jeux et programmes faisant appel au graphisme pour les ordinateurs personnels. Quant aux domaines scientifiques, les systèmes CAO (conception assistée par ordinateur) et DAO (dessin assisté par ordinateur) sont très répandus auprès des ingénieurs et architectes. On trouve une discussion plus élaborée sur les applications de l'informatique interactive graphique dans Foley et Van Dam (1984).

Du côté de la recherche opérationnelle, la puissance de la représentation graphique des réseaux a été suffisamment illustrée par plusieurs systèmes interactifs-graphiques pour la planification des réseaux de transport urbain de personnes. À ce sujet, soulignons les logiciels IGTDS (Interactive Graphic Transportation Design System, 1978) TNOP (Transit Network Optimization System, 1976), le système VOLVO (Andreasson, 1976; Hasselström, 1981) et EMME/2 (Spiess, 1984). Ce dernier présente également une excellente revue critique de ces systèmes.

À notre connaissance, il n'existe aucun système interactif graphique adapté aux besoins d'un instrument de planification tactique des réseaux de transport interurbain des marchandises. Toutefois, il existe quelques outils traitant les modèles de transport de marchandises au niveau national. Mentionnons le "Princeton Transportation Network Model and Graphic Information System" (Kornhauser, 1984), un ensemble imortant de programmes, banque de données et procédures interactives graphiques conçus pour l'analyse des flots unimodaux au niveau national et STAN (Florian, 1986), une méthode interactive graphique mise au point pour la planification stratégique du transport des marchandises au niveau national et régional.

On retrouve aussi cette lacune pour les problèmes de logistique tel la production/distribution des entreprises et la localisation des dépôts. Fisher (1985) dans son papier décrivant comment les systèmes homme/machine facilitent la définition, la révision et la résolution des modèles d'optimisation, présente deux approches interactives graphiques pour des modèles simples de localisation de dépôts. Quoique ces deux méthodes soient innovatrices et intéressantes, elles ne s'appliquent pas pour des modèles plus complexes telle la localisation de dépôts multiniveaux, multimodes, multiproduits, avec exigences d'équilibrage des charges des dépôts.

Depuis quelques années, nous travaillons à la réalisation d'un prototype interactif graphique intégrant un modèle et des algorithmes de recherche opérationnelle, pour la planification tactique du transport interurbain des marchandises. De plus, le noyau méthodologique de ce système possède plusieurs ressemblances avec l'approche utilisée pour la résolution du modèle de localisation énoncé plus haut. Malgré ces conformités, il n'est pas approprié de construire un logiciel interactif graphique intégrant les deux modèles (Crainic et Dejax, 1987).

Outre les principe généraux utilisés lors de la conception et du développement de ces systèmes (organisation modulaire, travail par scénario, interface par une combinaison dialogue-commandes graphiques), nous croyons que le seul noyau informatique commun soit une bibliothèque graphique de base pouvant s'adapter au graphisme de réseau. Pour ce genre de dessin, il faut distinguer les deux échelles suivantes : l'échelle cartésienne (emplacement des objets dans le plan) et l'échelle des attributs (taille des symboles dessinés). Deux discussions plus approfondies sur ce sujet se trouvent dans Spiess (1984) et Horowitz (1987).

Le présent document introduit une nouvelle bibliothèque graphique écrite en langage C. Quoiqu'on puisse utiliser BGC pour des applications graphiques simples (graphe XY, histogramme, etc.), elle est bien appropriée au graphisme de réseau. Nous commencerons par énoncer les concepts de base sous-jacents à la mise en oeuvre d'une bibliothèque graphique (chapitre 1). Nous continuerons en décrivant la sructure et l'environnement de BGC (chapitre 2) et nous terminerons par une description exhaustive des fonctions qui la composent (chapitre 3).

Nous soulignons que BGC est fortement inspiré de QPLOT (Spiess, 1984) et que cette première version supporte seulement les équipements Tektronix et leurs émulateurs.

Nous avons préféré le langage C car il est idéal pour la conception et l'application de projets de programmation complexes. Cette caractéristique est largement due à l'efficacité de son code, la concision des énoncés, le contrôle sur la représentation interne des données et leur durée de vie en mémoire ainsi que la grande souplesse que procure l'usage correct des fonctions. Nous soulignons que le C est un langage très "portable" et ses fonctions peuvent être appelées à l'intérieur d'un programme écrit dans un autre langage. Enfin, l'usage des pointeurs et structures permettent la manipulation efficace de structures de données évoluées.

1. CONCEPTS DE BASE

Une bibliothèque graphique est employée par un programme d'application de la même manière qu'un programme utilise les fonctions d'entrée-sortie du système d'exploitation. Ces fonctions d'entrée-sortie gèrent la communication entre le programme et l'architecture de bas niveau de l'ordinateur sans que le programmeur s'en soucie. De la même manière, les fonctions d'une bibliothèque graphique permettent l'utilisation des périphériques graphiques sans que le programmeur ait besoin de connaître leur architecture de bas niveau. La bibliothèque graphique est donc l'interface entre un programme d'application de haut niveau et l'équipement graphique.

Le schéma conceptuel d'un système interactif graphique est présenté à la figure 1.1. Il est composé d'un ensemble d'unités physiques (hardware) et d'un logiciel d'application (software).

La partie physique du système consiste en un ordinateur principal (host computer) auquel sont reliées des unités logiques d'entrée-sortie. Il y a cinq grandes classes d'unités logiques d'entrée :

KEYBOARD : clavier alphanumérique pour l'entrée des chaînes de caractères;

BUTTON : touches ou boutons programmables;

PICK : dispositifs pour pointer une information à l'écran (crayon lumineux);

LOCATORS : instruments qui retournent une coordonnée XY de l'écran (souris, joystick, tablette graphique, roulettes, écran tactile, etc.);

EVALUATORS : ce sont généralement des potentiomètres. Ils permettent de spécifier une valeur numérique.

Soulignons qu'il existe aussi d'autres méthodes d'entrée plus sophistiquées telle la reconnaissance de la voix. De plus, des algorithmes ont été élaborées pour simuler une classe logique à l'aide d'un dispositif d'une

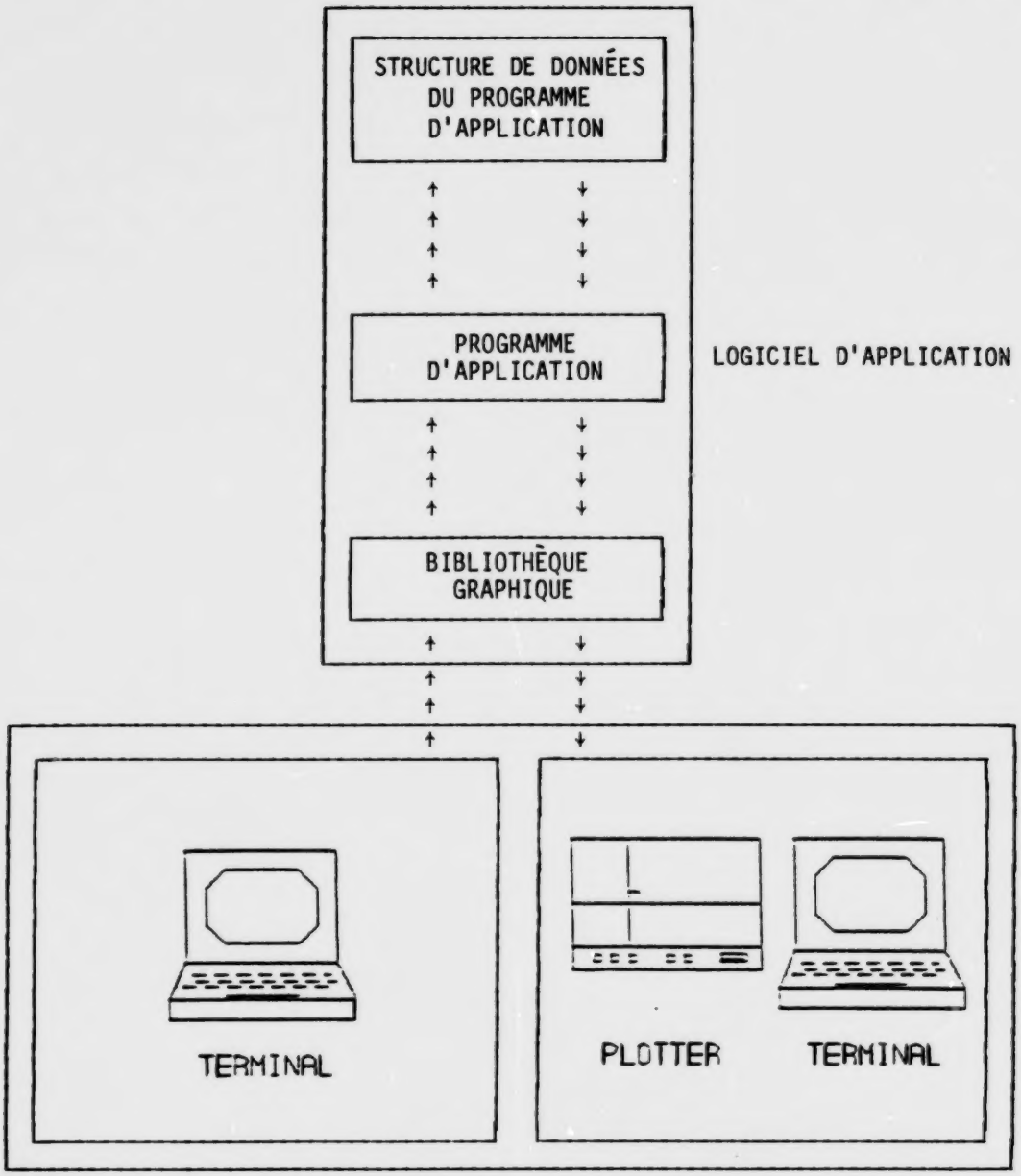

Figure 1.1 : Schéma conceptuel d'un système graphique

autre classe. Quant aux unités de sortie, elles génèrent des images sur une surface de visualisation (écran ou papier). Pour ces unités, on distingue deux classes logiques :

TERMINAL : moniteur graphique;

PLOTTER : dispositifs pour l'obtention des copies permanentes (hard-copy). On retrouve dans cette classe les tables traçantes, moniteurs reliés à un traceur graphique, etc.

Pour sa part, le logiciel est divisé en trois parties :

- structure de données du programme d'application
- programme d'application
- bibliothèque graphique

La structure de données contient la description des objets réels ou abstraits dont on veut une représentation graphique. Typiquement, un objet est défini par ses coordonnées dans un plan et des attributs décrivant sa taille et sa forme. On indique aussi le motif et la couleur des vecteurs avec lesquels l'objet sera dessiné. Enfin, des liens de connection entre les différents objets sont donnés pour spécifier leur relations.

Le programme d'application manipule les objets de la structure de données, gère les interactions usager-ordinateur et envoie les commandes à la bibliothèque graphique.

Enfin, la bibliothèque graphique contient un ensemble de fonctions, mises à la disponibilité du programme d'application, qui se divise en six grandes classes :

PRIMITIVES GRAPHIQUES : ce sont les fonctions qui permettent de dessiner des vecteurs, des rectangles, des cercles, etc. Ces procédures sont indépendantes de l'équipement graphique;

OPÉRATIONS DE VISUALISATION : ces fonctions effectuent les manipulations sur les fenêtres et les clôtures. De plus, elles accomplissent les opérations de découpage sur les vecteurs (clipping);

MODIFICATIONS DES ATTRIBUTS : par exemple, ces fonctions fixent la valeur courante du motif de tracé des vecteurs et de leur couleur;

ENTRÉE GRAPHIQUE : ce sont les fonctions qui permettent les entrées graphiques via la souris, la tablette graphique, etc.;

INITIALISATION ET CONTROLE : ce sont les fonctions qui initialisent la bibliothèque graphique;

INTERFACE DE BAS NIVEAU : ce sont les fonctions qui dépendent du matériel graphique. Elles envoient les commandes compatibles avec un standard graphique (Tektronix, IBM, POSTSCRIPT, etc.) pour tracer les vecteurs, activer une couleur, etc.

2. STRUCTURE ET ENVIRONNEMENT DE BGC

Notre bibliothèque graphique a été construite de manière à respecter les concepts de base mentionnés au chapitre précédent. De plus, nous avons organisé BGC comme une boîte noire, c'est-à-dire qu'elle est indépendante des programmes d'application qui l'utilisent.

La structure de BGC est illustrée à la figure 2.1 et elle est composée de quatre classes de fonctions :

- Primitives graphiques de haut niveau :
  - fonctions mathématiques
  - générateur de caractères
  - opérations de visualisation
  - vecteurs, rectangles, polygones, cercles;
- Interface avec l'équipement graphique :
  - entrée graphique
  - sélection des attributs (couleur, motif des traits)
  - commandes de bas niveau compatibles avec les unités graphiques (Tektronix pour cette version);
- Fonctions d'initialisation et de contrôle;
- Fonctions de sortie :
  - interface avec le système d'exploitation;

qui seront décrites plus en détail au chapitre suivant.

Nous commençons d'abord par décrire l'environnement nécessaire au fonctionnement de la bibliothèque graphique, c'est le répertoire graphique.

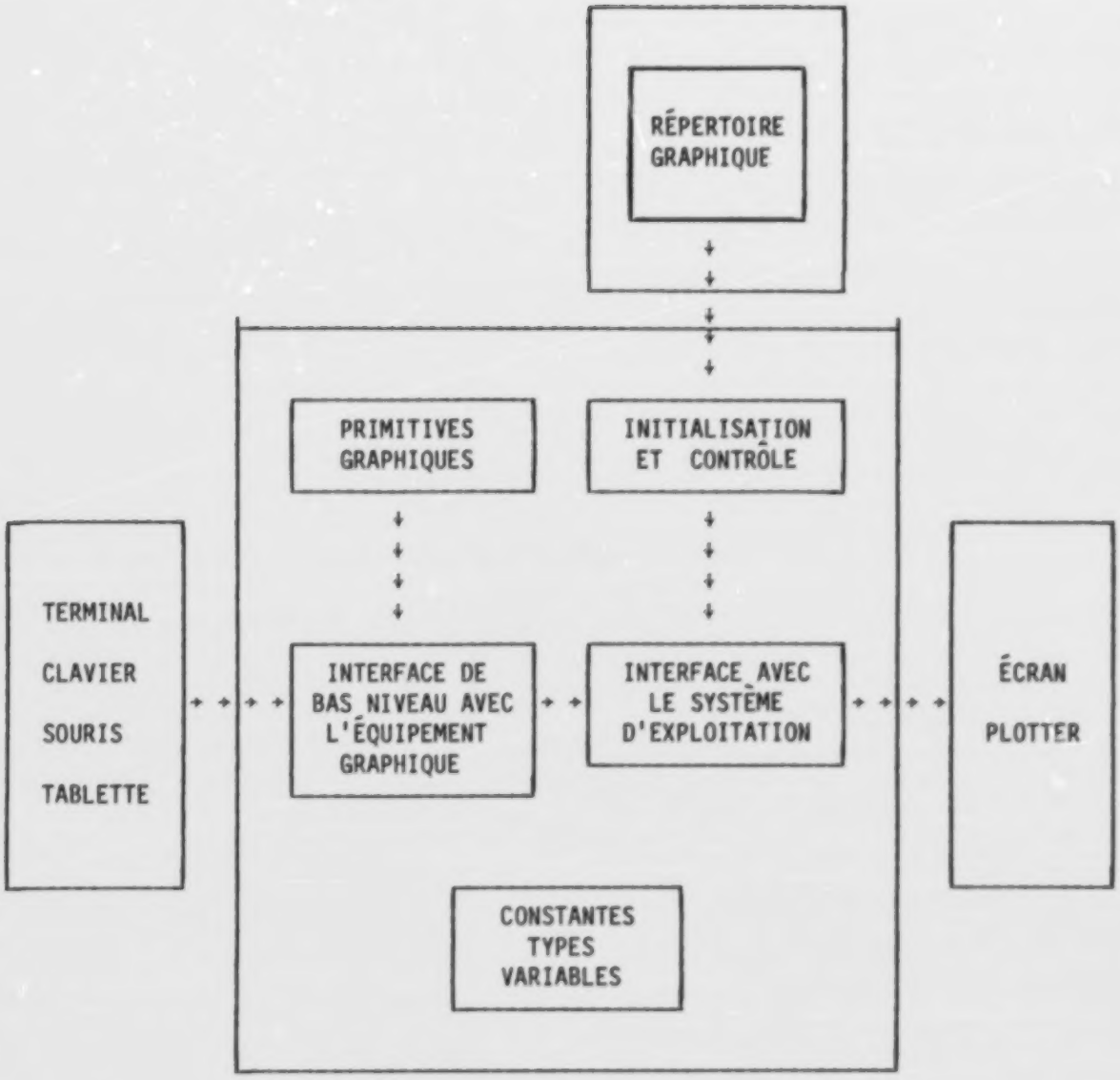

Figure 2.1 : Structure de BGC

2.1 Répertoire graphique

Pendant une session de travail graphique, BGC doit pouvoir lire ou écrire les données relatives aux informations suivantes :

2.1.1 Unité graphique courante

2.1.2 Tables d'initialisations

2.1.3 Fontes pour le générateur de caractères

2.1.4 Fenêtres sauvées sur disque

2.1.5 Dessins sauvés sur disque

Les fichiers qui contiennent ces informations sont réunis dans un seul répertoire défini par le programmeur d'application. C'est le répertoire graphique. En outre, BGC doit connaître le nom de ce répertoire. Cela s'effectue à l'intérieur du programme d'application en passant la chaîne de caractères correspondante à la fonction debut-session (voir fonctions d'initialisation).

2.1.1 Unité graphique courante

Au chapitre précédent, nous avions distingué deux classes d'unités logiques de sortie :

- TEMINAL
- PLOTTER

Pour BGC cela correspond à un canal de sortie. Les sorties graphiques sont envoyés à une seule classe à la fois. La classe qui les reçoit est la classe active. On remarque l'analogie entre ce principe et celui des entrées-sorties (input/output) du système d'exploitation UNIX. La classe TERMINAL correspond à la sortie standard (file descriptor = 1) tandis qu'on

fait une redirection quand on active la classe PLOTTER. La classe active par défaut est la classe TERMINAL. Le programmeur active une classe avec les fonctions active-terminal et active-plotter de BGC (voir fonctions d'initialisation).

Pour chacune des deux classes, la bibliothèque graphique reconnaît plusieurs appareils compatibles avec le standard graphique utilisé par l'interface de bas niveau. Une chaîne de caractères et un numéro ont été assignés à chaque unité graphique soutenue par BGC. Ce sont des constantes de compilation. Nous soulignons que cette façon de faire pose une restriction importante pour la modification de la liste des appareils reconnus par notre bibliothèque graphique. En effet, cela implique une recompilation des fonctions de BGC lorsqu'il y a un changement apporté à cette liste d'appareils. Par contre, il est difficile de faire autrement car les fonctions de l'interface de bas niveau doivent connaître a priori les identificateurs de ces appareils. Il existe malheureusement des émulateurs imparfaits des standard graphiques et les fonctions doivent contenir des exceptions (patch) pour les traiter. La fonction imp-version (3.1 fonctions d'intialisation et de contrôle) imprime à l'écran la liste des unités soutenues par BGC. De plus, on retrouve cette liste à l'annexe A lors de la description des contantes graphiques (termcap.h).

De plus, pendant une session de travail avec un programme d'application faisant appel à BGC, un seul appareil par classe reçoit les sorties graphiques. C'est l'unité graphique courante pour une classe de sortie donnée. L'unité graphique courante, pour chacune des deux classes doit être fixée avant l'utilisation du programme d'application . Il est impossible de modifier le choix d'une unité graphique courante pendant une session de travail avec le programme d'application. C'est le fichier CUR-DEV (current device) qui conserve l'unité graphique courante pour chacune des deux classes. C'est un fichier avec un seul enregistrement qui contient deux

entiers séparés par un blanc. Le premier représente le numéro de l'unité graphique courante de la classe de sortie TERMINAL et il doit correspondre à la constante de compilation associée au terminal utilisé lors de la séance de travail avec un programme d'application graphique. Le second entier représente l'appareil graphique de la classe de sortie PLOTTER. C'est la responsabilité de l'utilisateur du systèmes graphique de faire la mise à jour du fichier CUR-DEV avant une session de travail. Pour ce faire, il peut faire appel à l'utilitaire mcd décrit à l'annexe B. Une autre méthode est la modification du fichier avec un éditeur de texte. Cette dernière façon de faire est risquée car elle peut provoquer des erreurs de cohérence. C'est à l'usager de vérifier la concordance entre l'unité graphique courante d'une classe et l'appareil effectivement utilisé pour cette classe. Les erreurs de cohésion entraînent des effets imprévisibles lors de l'affichage (vecteur mal dessiné, couleur absente, caractères indésirables, etc.).

2.1.2 Tables d'initialisation

Puisque notre bibliothèque graphique supporte plusieurs dispositifs graphiques, elle doit connaître les chaînes d'initialisation et caractéristiques de chacun d'eux. Ces informations sont gardées dans des tables d'initialisation sous forme de fichiers. Il y a un fichier par unité graphique soutenue par BGC. Le nom d'un fichier contenant la table d'initialisation d'une unité graphique correspond au nom (chaîne de caractères) assigné à cette unité lors de la compilation (voir section 2.1.1). Pendant l'exécution du programme d'application, BGC garde en mémoire vive la table d'initialisation associée à l'unité graphique courante de la classe active.

Le format d'une table d'initialisation sous forme de fichier est donné à la figure 2.2 tandis que la figure 2.3 indique les caractéristiques d'une unité graphique.

ENREG.	DESCRIPTION	
1	Deux valeurs booléennes séparées par un blanc pour indiquer la classe de sortie d'appartenance de l'unité graphique. TERMINAL (V/F) PLOTTER (V/F) (Vrai = 1, Faux = 0) convention : un terminal graphique appartient aux deux classes.	DESCRIPTION DE L'UNITÉ GRAPHIQUE
2	Séquence de 16 bits décrivant les caractéristiques de l'unité graphique. Chaque bit est sépéré par un blanc. b_0 b_1 b_2 ... b_{15} (voir figure 2.3)	
3	Huit paramètres divers (voir figure 2.4) .1 EXTRA-BYTE .4 LINE-PAGE .7 MSEC-REP .2 NULL-CHAR .5 BAUD-RATE .8 MSEC-PLT .3 LINE-PATTERN .6 MSEC-DIAL	
4	Initialisation de l'unité graphique	CHAÎNES D'INITIALISATION
5	Initialisation des dialogues	
6	Initialisation alphanumérique	
7	Initialisation du curseur graphique	
8	Initialisation du mode graphique	
9	Fin de session graphique	

Figure 2.2 : Format d'une table d'initialisation

BIT #	DESCRIPTION
0	machine ASCII
1	Permet la sortie graphique
2	Permet l'entrée graphique
3	Vecteurs générés par l'appareil
4	Caractères générés (taille 1)
5	Caractères générés (taille 2)
6	Caractères générés (taille 3)
7	Caractères générés (taille 4)
8	Couleur disponible
9	Mode d'écriture graphique directe
10	Positionnement du curseur graphique
11	Effaçage sélectif
12..15	Pas encore utilisés

où $b_i = 1$ indique que la machine possède cette caractéristique

$b_i = 0$ sinon

Figure 2.3 : Caractéristique de l'unité graphique

NOM	DESCRIPTION
EXTRA-BYTE	Nombre de bytes supplémentaires à envoyer selon la résolution de l'unité graphique Unité graphique — EXTRA-BYTE Visual — 0 TEK 4110 SERIES — 1 TEK 4014 4015 — 1 TEK 4663 — 2
NULL-CHAR	Caractère nul
LINE-PATTERN	Nombre de motifs pour les vecteurs générés par l'unité graphique
LINE-PAGE	Nombre de lignes sur la surface de visualisation
BAUD-RATE	Vitesse du terminal
MSEC-DIAL MSEC-REP MSEC-PLT	Temps de réponse en milli-secondes

Figure 2.4 : Paramètres divers

Les tables d'initialisation sont placées dans le répertoire graphique où elles ne devraient pas être modifiées. Soulignons que les erreurs causées par la modification du contenu d'une table d'initialisation ne peuvent pas être détectées par les fonctions de la bibliothèque graphique.

2.1.3 Fichier des fontes

Ces fichiers contiennent la définition des caractères pour les différentes fontes définies par le générateur de caractères.

Fonte-0 : Définition de la fonte ASCII. Elle est chargée en mémoire vive pour la durée de la session graphique.

Fonte-1 : Créée par l'usager

Fonte-2 : Créée par l'usager

Chargée au besoin d'une manière dynamique.

Ref. publication "A Contribution to Computer Typesetting Techniques: Tables of Coordinates for Hershey's Repertory of Occidental Type Fonts and Graphic Symbols" par Norman M. Wolcott et Joseph Hilsenrath.

2.1.4 Fichier des fenêtres

C'est le fichier "WINDOW" qui conserve les différentes coordonnées des fenêtres qui ont été sauvées avec la fonction window (voir fonctions des primitives graphiques). Ce fichier est géré par BGC et il est créé au besoin s'il n'existe pas.

2.1.5 Fichier des dessins

Les sorties graphiques envoyées à la classe PLOTTER sont emmagasinées dans le fichier des dessins pour une utilisation ultérieure (table traçante, moniteur relié à un traceur graphique screen dump), imprimante, etc.). Le nom de ce fichier est "PLOTS"et, comme pour le fichier des fenêtres, il est géré par la bibliothèque graphique.

3. DESCRIPTION DES FONCTIONS DE BGC

Dans ce dernier chapitre, nous présentons les 56 fonctions de BGC mises à la disposition des programmeurs de systèmes graphiques. Elles sont regroupées par classe logique des tâches qu'elles effectuent et nous les décrirons aux sections et sous-sections selon le tableau 3.1.

De plus, le format de présentation utilisé ressemble à celui employé par les manuels d'instruction du système d'exploitation UNIX. Chaque page contient la description d'une ou plusieurs fonctions selon leur degré d'homogénéité. Nous commençons par donner le nom ou les noms des fonctions. Ensuite, nous présentons la ou les fonctions et leurs paramètres dans le format du langage C en utilisant des figures au besoin.

TABLEAU 3.1 : Listes des fonctions de BGC

3.1 Fonctions d'initialisation et de contrôle

3.1.1 Initialisation

début-session (rep)
fin-session ( )
active-plotter ( )
active-terminal ( )

3.1.2 Contrôle

cloche (i)
efface-écran ( )
imp-version ( )
obt-date ( )
obt-motif ( )
obt-couleur ( )
obt-taille ( )
obt-fonte ( )
obt-gross ( )
obt-lg-chaîne ( )

3.2 Fonctions de sorties

env-ascii (ascii)
vider-tampon ( )
env-chaîne (chaîne,nb-car,fac-rep)

3.3 Interface graphique de bas niveau

3.3.1 Génération de vecteurs

point-abs (x,y)
point-rel (dx,dy)
ligne-abs (x,y)
ligne-rel (dx,dy)

TABLEAU 3.1 : Listes des fonctions de BGC (suite)

3.3.2 Modification des attributs

fixe-couleur (couleur)
fixe-fonte (fonte)
fixe-gross (fac-gross)
fixe-motif (motif)
fixe-taille (taille)

3.3.3 Entrée graphique

lire-locator (x,y,signal)

3.3.4 Effaçage sélectif

eff-rec (x1,y1,x2,y2)

3.3.5 Caractères générés par l'unité graphique

ecrit-chaîne (x,y,chaîne,nbcar)
ecrit-nombre (x,y,nombre,format)

3.4 Fonctions de primitives graphiques

3.4.1 Opération de visualisation

def-clôture (x1,y1,x2,y2,aj)
def-fenêtre (x1,y1,x2,y2,aj)
obt-clôture (x1,y1,x2,y2)
obt-fenêtre (x1,y1,x2,y2)
conv-clôture (xf,yf,xc,yc)
conv-fenêtre (xc,yc,xf,yf)
window (com)

TABLEAU 3.1 : Listes des fonctions de BGC (suite)

3.4.2 Primitives graphiques de haut niveau

dess-arc (xc,yc,xd,yd,syst-coor,longueur,nb-seg)
dess-aster (xc,yc,syst-coor,rayon)
dess-cercle (xc,yc,syst-coor,rayon,nb-seg)
dess-échelle (x,y,e)
dess-flèche (xd,yd,xf,yf,syst-coor,b,h)
dess-lien (x1,y1,x2,y2,t1,t2,d,e,v,c)
dess-noeud (x,y,t,mn,nu,c)
dess-rec (xc,yc,syst-coor,b,h)
dess-vec (x1,y1,x2,y2,syst-coor)
joint-cercle (x1,y1,x2,y2,syst-coor,r1,r2,d)
joint-rec (x1,y1,x2,y2,syst-coor,b1,h1,b2,h2,d)

3.4.3 Générateur de caractères

dess-ch (x1,y1,x2,y2,syst-coor,chaîne,nb-car)
dess-ch-vec (x1,y1,x2,y2,syst-coor,d,chaîne,nb-car)
dess-nb-vec (x1,y1,x2,y2,syst-coor,d,nombre,format)

3.4.4 Opérations mathématiques

dep-vec (x1,y1,x2,y2,x1d,y1d,x2d,y2d,syst-coor,d)
equat-droute (x1,y1,x2,y2,pente,bêta)
inter-vec (xa1,ya1,xa2,ya2,xb1,yb1,xb2,yb2,yi,yi)
inter-cer-vec (x1,y1,x2,y2,r,d,xi,yi)
inter-rec-vec (x1,y1,x2,y2,xc,yc,b,h,xi,yi)

3.1 Fonctions d'initialisation et de contrôle

Cette section se divise en deux sous-sections :

3.1.1 Initialisation

début-session (rep)
fin-session ( )
active-plotter ( )
active-terminal ( )

3.1.2 Contrôle

cloche (i)
efface-écran ( )
imp-version ( )
obt-date ( )
obt-motif ( )
obt-couleur ( )
obt-taille ( )
obt-fonte ( )
obt-gross ( )
obt-lg-chaîne ( )

3.1.1 Fonctions d'initialisation

Les fonctions regroupées dans cette section permettent l'initialisation de la bibliothèque graphique sur les deux unités graphiques courantes. La figure 3.1 illustre l'utilisation de ces fonctions à l'intérieur d'un programme d'application.

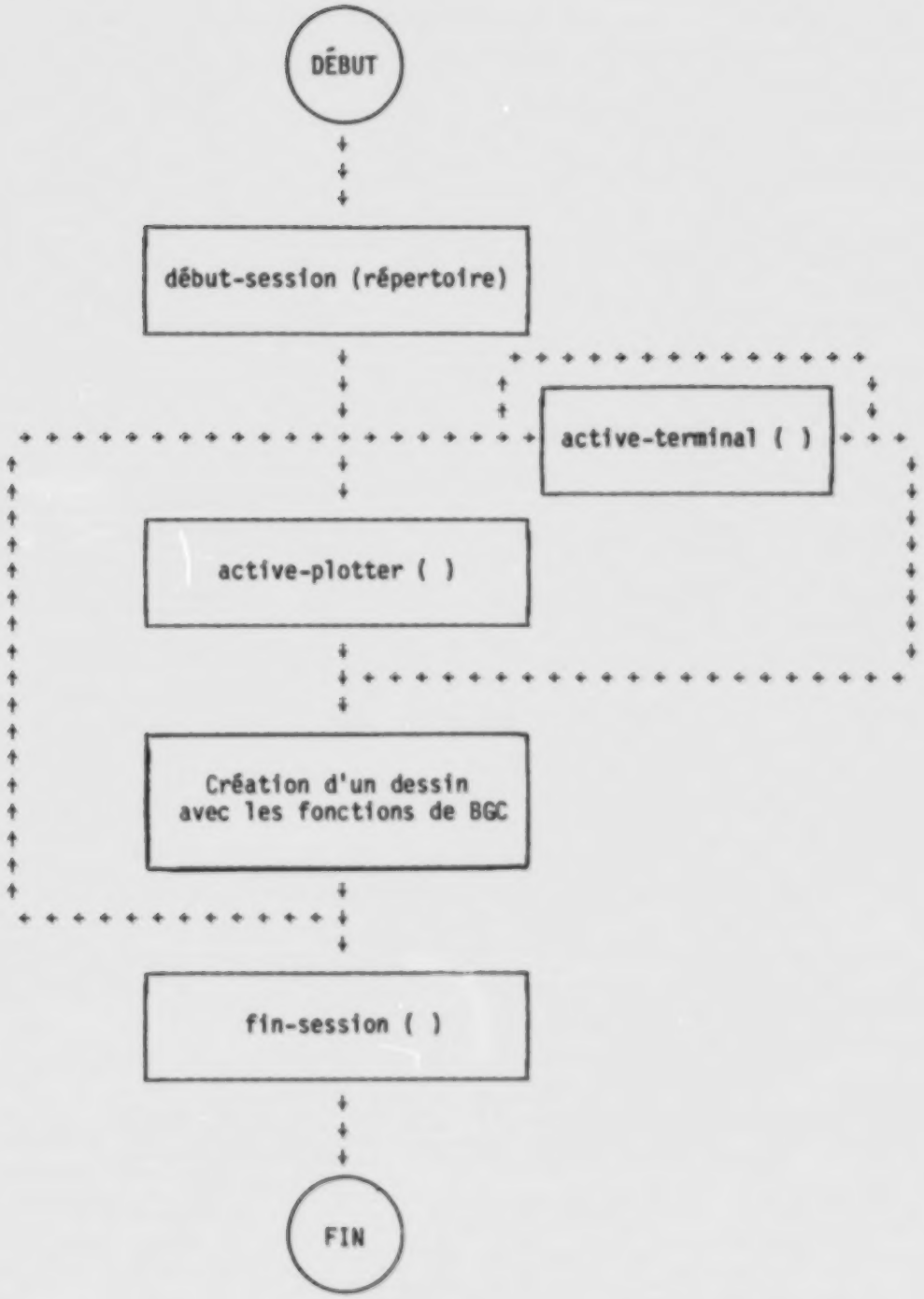

Figure 3.1 : Utilisation des fonctions d'initialisation

NOM

début-session

SYNOPSIS

```
void début-session (répertoire)
char *répertoire ; /* chaîne de caractères désignant le nom du */
                   /* répertoire graphique                     */
                   /* voir section 2.1                         */
```

DESCRIPTION

C'est la fonction de base pour débuter une session graphique. À ce moment, le répertoire graphique est localisé, l'unité graphique courante pour chacune des deux classes est identifié, les tables d'initialisation correspondantes sont lues. La fonte ASCII utilisée par le générateur de caractères est chargée en mémoire vive et les fichiers des dessins et des fenêtres sont ouverts ou créés. Soulignons que BGC assume que le fichier "PLOTS" est le premier fichier ouvert en permanence.

Le programmeur d'application doit s'assurer qu'il n'y a aucun fichier ouvert avant l'appel à cette fonction. Dans le cas contraire, il peut en perdre le contenu. Au commencement, la classe TERMINAL est active par défaut.

NOM

fin-session

SYNOPSIS

void fin-session ( )

DESCRIPTION

Cette fonction termine une session graphique en fermant les fichiers ouverts par la fonction début-session, en effaçant et en mettant le terminal en mode alphanumérique.

NOM

active-plotter, active-terminal

SYNOPSIS

void active-plotter ( )

void active-terminal ( )

DESCRIPTION

active-plotter redirige les sorties graphiques vers le fichier des dessins ("PLOTS"), selon les caractéristiques de l'unité graphique courante de la classe PLOTTER. Cette classe restera active jusqu'au prochain appel à la fonction active-terminal.

La fonction active-terminal redonne le contrôle au terminal courant en déconnectant le fichier "PLOTS" de la sortie graphique.

3.1.2 Fonctions de contrôle

Ces fonctions permettent au programmeur d'utiliser les fonctions du système d'exploitation pour faire sonner la cloche, obtenir la date, effacer l'écran, etc. Il y a aussi une fonction qui imprime la version de BGC et donne la liste des appareils graphiques. Enfin, cette sous-section regroupe aussi les fonctions qui retournent la valeur des attributs.

NOM

cloche, efface-écran, imp-version

SYNOPSIS

```
void cloche (i)
int i ;

void efface-écran ( )

void imp-version ( )
```

DESCRIPTION

La fonction cloche fait sonner la cloche du terminal i fois.

La fonction efface-écran efface et met le terminal en mode alphanumérique.

La fonction imp-version imprime à l'écran la version de la bibliothèque grapique et donne la liste des appareils graphiques soutenus par cette version.

```
Exemple : BGC version 1.0
          1 TK4014
          2 VIS500
          3 VIS550
          4 TK4111
          5 TK4115
          6 TK4663
```

NOM

obt-date

SYNOPSIS

char *obt-date ( )

DESCRIPTION

Cette fonction retourne un pointeur vers un tableau de 26 caractères ASCII qui contient la date et l'heure. Le format de ce tableau est :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
F	R	I		S	E	P		2	5		1	2	:	1	4	:	5	2		1	9	8	7	\n	\0

NOM

obt-motif, obt-couleur, obt-taille, obt-fonte, obt-gross, obt-lg-chaîne

SYNOPSIS

```
int obt-motif   ( ) /*Retourne la valeur courante du motif de*/
                    /*tracé des vecteurs                      */

int obt-couleur ( ) /*Retourne la valeur courante de la couleur*/
                    /*de tracé des vecteurs                   */

int obt-taille  ( ) /*Retourne la valeur courante de la taille des*/
                    /*caractères générés par l'unité graphique    */

int obt-fonte   ( ) /*Retourne la valeur courante de la fonte */
                    /*utilisée par le générateur de caractères*/

float obt-gross ( ) /*Retourne la valeur courante du facteur de*/
                    /*grossissement des caractères dessinés par*/
                    /*le générateur                           */

float obt-lg-chaîne (chaîne)
char *chaîne ;
                    /*Retourne la longueur d'une chaîne*/
                    /*(dessinée par le générateur de   */
                    /*caractères) en nombre de pixels  */
                    /*de l'écran normalisé             */
```

RÉFÉRENCE

Voir section 3.3.2, modification des attributs.

3.2 Fonctions de sortie

Les fonctions comprises dans cette section font l'interface entre BGC et le système d'exploitation. Elles effectuent les opérations d'envoie des caractères à l'unité graphique courante de la classe active. Ils peuvent être des caractères imprimables ou des caractères de contrôle. Dans ce dernier cas, ils sont utilisés pour activer les différentes fonctions des dispositifs physiques telle la cloche, le curseur graphique, etc.

Les fonctions de sortie sont :

env-ascii (ascii)
vider-tampon ( )
env-chaîne (chaîne,nb-car,fac-rep)

NOM

env-ascii, vider-tampon

SYNOPSIS

```
void env-ascii (ascii)
char ascii ; /*Caractère ASCII à envoyer        */
             /*note : 'a' et 64 sont équivalents*/
void vider-tampon ( )
```

DESCRIPTION

La fonction env-ascii emmagasine dans un tampon le caractère reçu. La taille de ce tampon est fixée par la constante BUFFER-SIZE (voir Annexe A, fichier c.graph-h). Lorsque ce tampon est plein, il est envoyé automatiquement à l'unité graphique active par la fonction env-chaîne.

La fonction vider-tampon permet de vider le tampon de caractères mentionné ci-haut à l'unité graphique courante de la classe active.

NOM

env-chaîne

SYNOPSIS

```
void env-chaîne (chaîne, nb-car, fac-rep)

char *chaîne ; /*Chaîne de caractères à envoyer */
               /*terminée par le caractèeres NUL*/

int  nb-car  ; /*Nombre de caractères à envoyer*/

int  fac-rep ; /*Nombre de fois que la chaîne*/
               /*est envoyée                 */
```

DESCRIPTION

C'est la fonction de base pour les opérations de sortie car elle fait le lien entre la bibliothèque graphique et le système d'exploitation. Par cette fonction, la chaîne de caractères est envoyée directement (sans tampon, "unbuffered") à la sortie graphique active.

3.3 Interface graphique de bas niveau

Les fonctions énumérées dans cette section effectuent les opérations d'interface entre les primitives graphiques et les appareils TEKTRONIX ou compatibles. Elles envoient les commandes aux unités graphiques pour générer des vecteurs (section 3.3.1), modifier leurs attributs (section 3.3.2) et activer les dispositifs d'entrée graphique (section 3.3.3). De plus, elles permettent d'utiliser les caractéristiques de ces unités, tel l'effaçage sélectif (section 3.3.4), et les caractères générés par celles-ci (section 3.3.5).

Pour ces fonctions, nous avons fixé arbitrairement un écran graphique normalisé dont les dimensions sont les suivantes :

$$0.0 < x < 1024.0$$
$$0.0 < y < 780.0$$

Ce système de coordonnées logique recouvre toute la surface de visualisation des unités graphiques employées et possède l'avantage d'être indépendant de ceux-ci. Cet écran définit la clôture ("viewport") maximale qui est utilisée pour les opérations de visualisation. C'est l'univers graphique TEKTRONIX. De plus, le choix de ces dimensions ne limite pas la précision du dessin car elles admettent des valeurs réelles. C'est à la dernière étape qu'elles sont transormées et tronquées pour tenir compte de la résolution de l'équipement.

3.3.1 Génération de vecteurs

point-abs (x,y)

point-rel (dx,dy)

ligne-abs (x,y)

ligne-rel (dx,dy)

NOM

point-abs, point-rel

SYNOPSIS

```
void point-abs (x,y)
float x,y ;              /*0.0 ≤ x ≤ 1024.0*/
                         /*0.0 ≤ y ≤  780.0*/

void point-rel (dx,dy)
float dx,dy ;            /*0.0 ≤ pc_x+ dx ≤ 1024.0*/
                         /*0.0 ≤ pc_y+ dy ≤  780.0*/
```

DESCRIPTION

Ces deux fonctions permettent de positionner le point courant ou la plume graphique. L'adresse du point courant est (pc_x, pc_y). ligne-abs est un positionnement absolu. Dans ce cas $(pc_x, pc_y) \leftarrow (x,y)$.

ligne-rel effectue un déplacement de la plume graphique à partir du dernier point courant. Cela correspond à l'affectation $(pc_x, pc_y) \leftarrow (pc_x + dx, pc_y + dy)$.

NOM

ligne-abs, ligne-rel

SYNOPSIS

```
void ligne-abs (x,y)
float x,y ;              /*0.0 ≤ x ≤ 1024.0*/
                         /*0.0 ≤ y ≤  780.0*/

void ligne-rel (dx,dy)
float dx,dy ;            /*0.0 ≤ x + pcx ≤ 1024.0*/
                         /*0.0 ≤ y + pcy ≤  780.0*/
```

DESCRIPTION

Ces deux fonctions permettent de tracer un vecteur à partir du dernier point courant. Il n'y a aucun découpage et les coordonnées sont spécifiées en unités de l'écran normalisé. De plus, le motif de tracé des vecteurs est celui qui a été fixé par la fonction fixe-motif (voir section 3.3.2).

ligne-abs trace une ligne à partir du point courant jusqu'au point de coordonné (x,y). Aussi, le point courant prend la valeur (x,y).

ligne-rel dessine une ligne qui relie les points dont les coordonnées sont (pc_x,pc_y) et (pc_x+ dx, pc_y+ dy). De plus, le point courant est déplacé à l'adresse (pc_x+ dx, pc_y+ dy).

3.3.2 Modification des attributs

fixe-couleur (couleur)
fixe-fonte (fonte)
fixe-gross (fac-gross)
fixe-motif (motif)
fixe-taille (taille)

NOM

fixe-couleur

SYNOPSIS

```
void fixe-couleur (couleur)
short couleur ;     /*0 < couleur < 15, sinon la dernière*/
                    /*couleur reste active                */
```

DESCRIPTION

Lorsque l'unité graphique le permet, cette fonction fixe la nouvelle couleur de tracé des vecteurs. Cette version de BGC utilise les couleurs offertes par les terminaux TEKTRONIX couleurs. Le tableau suivant donne la liste des seize couleurs disponibles avec les constantes correspondantes définies dans le fichier c.graph.h.

COULEUR	CONSTANTE	VALEUR DU PARAMÈTRE D'ENTRÉE
Noir	BLEU	0
Blanc	BLANC	1
Rouge	ROUGE	2
Vert	VERT	3
Bleu	BLEU	4
Cyan	CYAN	5
Magenta	MAGENTA	6
Jaune	JAUNE	7
Orange	ORANGE	8
Vert jaune	VERT-J	9
Vert cyan	VERT-C	10
Bleu cyan	BLEU-C	11
Bleu magenta	BLEU-M	12
Rouge magenta	ROUGE-M	13
Gris foncé	GRIS-F	14
Gris pâle	GRIS-P	15

NOM

fixe-fonte

SYNOPSIS

```
void fixe-fonte (fonte)
short fonte ;
```

DESCRIPTION

Cette fonction permet de fixer la fonte qui sera utilisée par le générateur de caractères. Par défaut, la génération des caractères s'effectue avec la fonte ASCII. Le tableau suivant résume les actions prises par cette fonction.

VALEUR	CONSTANTE	ACTION
0	ASCII	La fonte ascii est activée pour le générateur de caractères.
1..9		La fonte correspondante est activée pour le générateur de caractères. De plus, elle est chargée dynamiquement en mémoire vive si elle ne l'était pas déjà. S'il y avait une autre fonte en mémoire dynamique, cette dernière serait libérée.
-1	LIBÈRE	La fonte chargée en mémoire dynamique est libérée et la fonte ascii est activée.

NOM

fixe-gross

SYNOPSIS

```
void fixe-gross (fac-gross)
float fac-gross     /*fac-gross > 0.0*/
```

DESCRIPTION

Pour fixer la taille des caractères dessinés par le générateur de caractères. La figure 3.2 illustre quelques valeurs possibles pour le paramètre d'entrée. Par défaut, le facteur de grossissement est 0.4.

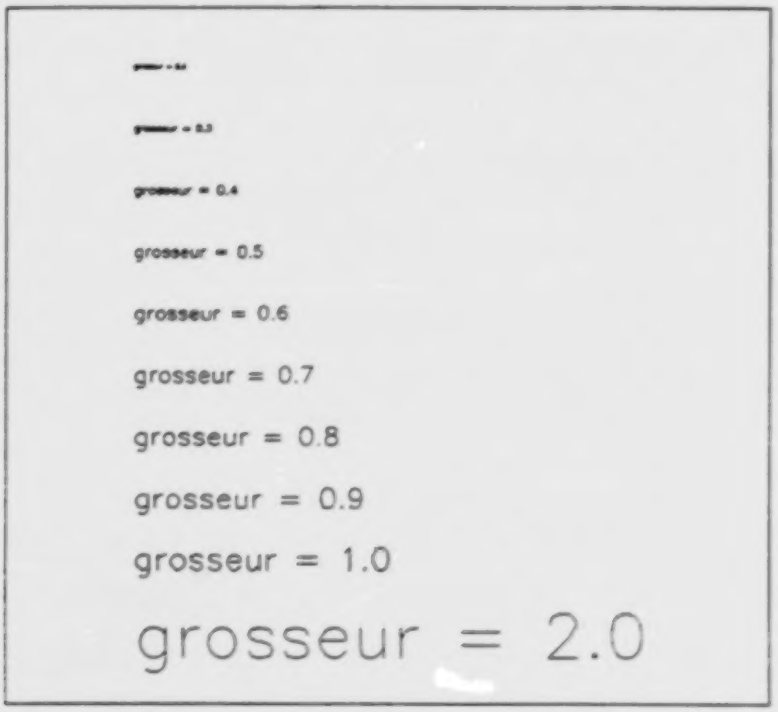

Figure 3.2 : Exemples de facteurs de grossissement

NOM

fixe-motif

SYNOPSIS

```
void fixe-motif (motif)
short motif ;          /*0 < motif < 5, sinon le dernier motif*/
                       /* demeure actif                       */
```

DESCRIPTION

Cette fonction fixe le nouveau motif de tracé des vecteurs. Les motifs disponibles sont ceux définis pour les équipements TEKTRONIX et sont illustrés au tableau suivant.

Tableau 3.2

Description des motifs de tracé des vecteurs

DESCRIPTION	CONSTANTE (c-graph.h)	VALEUR DU PARAMÈTRE	ILLUSTRATION
invisible	INVISIBLE	0	
visible plein	PLEIN	1	————
visible pointillé	POINT	2	.....
visible pointillé-trait	PT-TRAIT	3	—·—·—·—
visible à petits traits	TE-TRAIT	4	- - - - -
visible à longs traits	LG-TRAIT	5	— — — —

NOM

fixe-taille

SYNOPSIS

```
void fixe-taille (taille)
short taille ;          /* 0 < taille < 4, sinon la taille*/
                        /* précédente reste inchangée     */
```

DESCRIPTION

Cette fonction permet de fixer la taille des caractèeres générés par l'unité graphique (hardware generated character). Les unités graphiques TEKTRONIX ont quatre grosseurs disponibles de caractères. Les valeurs possibles pour le paramètre d'entrée sont données dans le tableau suivant. Les autres valeurs sont ignorées et la dernière taille reste active.

Tableau 3.3

Taille des caractères générés par l'unité graphique

TAILLE	LARGEUR	HAUTEUR	ESP. HOR.	ESP. VERT	Nombre de caractères par ligne	Nombre de lignes
	en nombre de pixels de l'écran normalisé					
1	10.00	16.00	14.00	22.00	75	35
2	8.50	14.00	12.75	20.50	80	38
3	6.00	10.00	8.50	13.25	120	58
4	5.00	8.00	7.75	12.00	132	65

3.3.3 Entrée graphique

Pour l'instant, il y a une seule fonction qui effectue l'entrée graphique. Cette fonction est lire-locator.

NOM

lire-locator

SYNOPSIS

```
void lire-locator (x,y,signal)
float *x, *y ;              /*Position du curseur au moment de*/
                            /*l'interruption                  */
                            /*0.0 < *x < 1024.0               */
                            /*0.0 < *y <  779.0               */
```

DESCRIPTION

Lorsque cette fonction est appelée, un curseur graphique qui peut être manipulé à l'aide d'une souris, d'un joystick ou des roulettes apparaît à l'écran. À ce moment, le système est en attente d'une interruption. Celle-ci se produit quand une touche du clavier ou un bouton du mécanisme d'entrée est pressé. Après l'interruption la bibliothèque graphique reçoit la position du curseur en coordonnées normalisées de l'écran et le signal qui l'a provoqué (code ACSII du caractère tapé ou caractère de contrôle).

3.3.4 Effaçage sélectif

NOM

eff-rec

SYNOPSIS

```
void eff-rec (x1,y1,x2,y2)
float x1,y1 ;              /*Coin inférieur gauche du rectangle*/
      x2,y2 ;              /*Coin supérieur droit du rectangle */
```

DESCRIPTION

Lorsque le terminal offre la possibilité d'effacer sélectivement une portion de l'écran ("selective erase"), cette fonction couvre un rectangle avec la couleur du fond ("background color"). Sinon, tout l'écran est effacé.

Le rectangle est défini par ses coins inférieur gauche et supérieur droit en coordonnées normalisées. Un rectangle partiellement ou complètement à l'extérieur de la clôture maximale est ignoré.

3.3.5 Caractères générés par l'unité graphique

Il y a deux fonctions décrites dans cette sous-section. Ce sont les fonctions :

écrit-chaîne (x1,y1,chaîne,nb-car)

écrit-nombre (x1,y1,nombre,format)

NOM

écrit-chaîne,écrit-nombre

SYNOPSIS

```
void écrit-chaîne (x1,y1,chaîne,nb-car)

float x1,y1  ;          /*0.0 < x1 < 1024.0*/
                        /*0.0 < y1 <  780.0*/
char *chaîne ;
short nb-car ;

void écrit-nombre (x1,y1,nombre,format)

float x1,y1  ;
float nombre ;
char *format ;          /*Format d'écriture d'un réel en C*/
                        /*"%3.1f","%5.0f", etc.          */
```

DESCRIPTION

La chaîne de caractères est imprimée horizontalement en commençant à l'adresse (x1,y1). La fonte utilisée est celle générée par l'équipement graphique ("hardware generated character"). La grosseur des caractères correspond à la taille définie avec la fonction fixe-taille. Par défaut, la taille 4 est utilisée.

écrit-nombre code le nombre réel dans sa chaîne de caractères correspondante. Cette dernière est ensuite envoyée à l'unité graphique avec la fontion écrit-chaîne.

3.4 Primitives graphiques

C'est le niveau hiérarchique le plus élevé de BGC. Ce sont des fonctions indépendantes du système d'exploitation et des équipements graphiques. Elles permettent les opérations de visualisation (section 3.4.1), de création de primitives graphiques de haut niveau (section 3.4.2) et de génération de caractères (section 3.4.3). Elles effectuent aussi des opérations mathématiques d'intersection, de déplacement et de découpage sur des vecteurs (section 3.4.4).

3.4.1 Opérations de visualisation

Les objets à afficher sont définis par des coordonnées arbitrairement choisies par le programmeur. BGC doit projeter ces coordonnées dans l'univers de l'écran normalisé pour obtenir un dessin à l'unité graphique. Pour effectuer cette conversion, BGC doit connaître la portion du système de coordonnées du programmeur qui contient l'information à être dessinée à ce moment. Cette région rectangulaire de l'univers de l'usager est une fenêtre.

Il ne faut pas confondre cette terminologie dans le contexte graphique avec la définition présentée dans la documentation des interfaces orientées-écran. Pour ces systèmes, la fenêtre désigne une région de l'écran. Pour la bibliothèque graphique, une clôture ("viewport") est une région rectangulaire de l'écran normalisé dans laquelle les informations de la fenêtre sont affichées.

NOM

obt-clôture,obt-fenêtre

SYNOPSIS

```
void obt-clôture (x1,y1,x2,y2)

float *x1,*y1 ;          /*Coordonnée du coin inférieur gauche*/
      *x2,*y2 ;          /*Coordonnée du coin supérieur droit */
                         /*de la clôture                      */

void obt-fenêtre (x1,y1,x2,y2)

float *x1,*y1 ;          /*Coordonnée du coin inférieur gauche*/
      *x2,*y2 ;          /*Coordonnée du coin supérieur droit */
                         /*de la fenêtre                      */
```

DESCRIPTION

Ces deux fonctions permettent d'obtenir les coordonnées de la clôture et de la fenêtre courante.

RÉFÉRENCES

def-clôture,def-fenêtre

NOM

conv-clôture,conv-fenêtre

SYNOPSIS

```
void conv-clôture (x-fen,y-fen,x-clot,y-clot)

float x-fen,y-fen     ;
float *x-clot,*y-clot ;      /Résultat de la projection    */

int conv-fen (x-clot,y-clot,x-fen,y-fen)

float x-clot,y-clot ;
float *x-fen,*y-fen ;        /Résultat de la transformation*/
```

DESCRIPTION

conv-clôture transforme une coordonnée de la fenêtre en un point du système de coordonnées de la clôture courante. Cette fonction renvoie la valeur 1 si la coordonnée (x-fen,y-fen) est à l'intérieur de la fenêtre courante. Sinon la valeur de retour est 0.

La seconde fonction convertit une coordonnée de la clôture (écran normalisé) dans une coordonnée de la fenêtre courante. Les valeurs de retour sont 1 si la coordonnée (x-clot,y-clot) est à l'intérieur de la clôture courante et 0 sinon.

NOM

def-clôture

SYNOPSIS

```
int def-clôture (x1,y1,x2,y2,aj)
float x1,y1 ;          /*Coordonnée du coin inférieur          */
                       /*Coordonnée du coin supérieur droit    */
                       /*de la nouvelle clôture                */

boolean aj ;           /*TRUE = Ajustement des coordonnées pour*/
                       /*conserver les ratios en x et y        */
                       /*avec la fenêtre courante              */
```

DESCRIPTION

Cette fonction permet de définir la clôture courante à l'intérieur de l'écran normalisé. Par défaut, la clôture courante couvre tout l'écran normalisé. Lorsque la nouvelle clôture est partiellement ou complètement à l'extérieur du système de coordonnées normalisées, celle-ci est ignorée et la clôture précédente reste active. Soulignons que les coordonnées de la clôture précédente ne sont pas conservées quand la nouvelle clôture est activée.

Cette fonction retourne les valeurs suivantes :

1 si la clôture est acceptée

0 sinon

NOM

def-fenêtre

SYNOPSIS

```
void def-fenêtre (x1,y1,x2,y2,aj)
float x1,y1 ;          /*Coordonnée du coin inférieur gauche    */
      x2,y2 ;          /*Coordonnée du coin supérieur droit     */
                       /*de la nouvelle fenêtre                 */

boolean aj ;           /*TRUE = Ajustement des coordonnées de la*/
                       /*fenêtre pour conserver les ratios en   */
                       /*x et y avec la clôture                 */
```

DESCRIPTION

Cette fonction permet de définir la fenêtre courante. Il n'y a aucune restriction sur les coordonnées et la fenêtre par défaut correspond à l'écran normalisé. Comme précédemment, lorsqu'une nouvelle fenêtre est spécifiée, les coordonnées de l'ancienne sont perdues. Il n'y a pas de valeur de retour pour cette fonction.

NOM

window

SYNOPSIS

```
void window (com)
short com ;              /*1 < com < 11    */
                         /*voir tableau 3.4*/
```

DESCRIPTION

Cette fonction met à la disposition du programmeur des commandes avec lesquelles l'utilisateur peut créer et manipuler des fenêtres interactivement.

Tableau 3.4

Valeurs possibles pour le paramètre COM

Paramètre d'entrée (COM)	Constante c-graph-h	Description
1	CURSEUR	Les coins inférieur gauche et supérieur droit sont pointés à l'aide du curseur graphique.
2	CENTRE	La nouvelle fenêtre garde la même dimension que la précédente mais elle est centrée au point indiqué par le curseur graphique.
3	ZOOM	L'utilisateur doit taper au clavier la valeur de ZOOMING. 0.0 < ZOOM < 1.0 → ZOOM IN 0.0 < ZOOM → ZOOM OUT
4	COORD	L'usager est appelé à spécifier les coordonnées de la nouvelle fenêtre avec le clavier numérique.
5	DROITE	Voir la figure 3.3
6	GAUCHE	
7	HAUT	
8	BAS	
9	DERNIÈRE	La fenêtre qui est à la tête de la liste circulaire est activée.
10	SAUVER	Permet d'enregistrer sur le disque les coordonnées de la fenêtre courante. L'usager peut lui assigner un identificateur d'au plus dix caractères.
11	RAPPELER	Pour activer une fenêtre sauvée sur le disque.

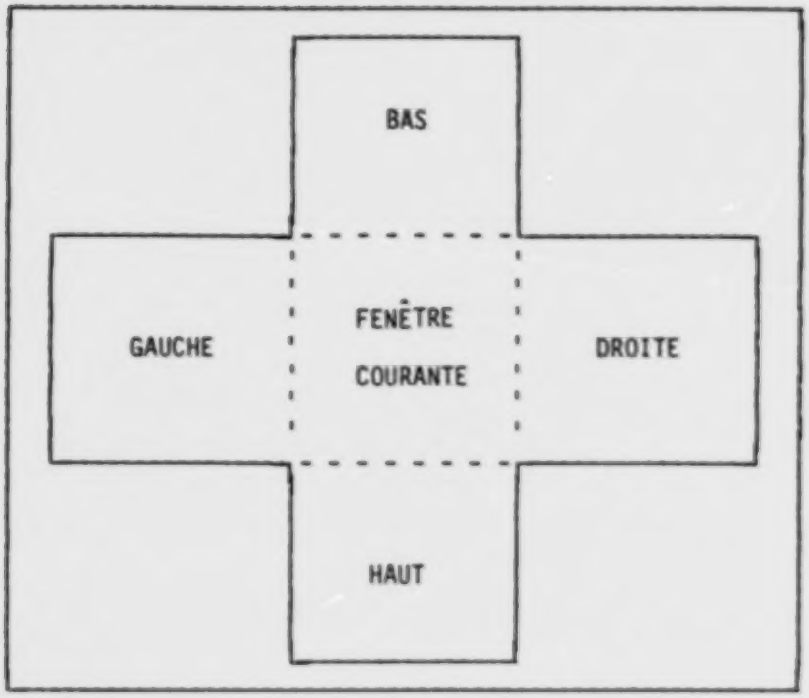

Figure 3.3 : Fenêtres spécifiées par les commandes 5, 6, 7 et 8

Il y a une liste circulaire qui emmagasine les dix dernières fenêtres définies avec cette fonction. Lorsqu'une nouvelle fenêtre est spécifiée, elle est activée avec la fonction def-fenêtre tandis que la précédente est placée dans la liste circulaire.

Le tableau 3.4 résume les commandes disponibles. Pour chacune d'elle, nous donnons la valeur du paramètre d'entrée à la fonction window pour l'invoquer, la constante qui peut être utilisée par le programme d'application et une brève description.

Toutes les commandes énumérées ci-bas, sauf les commandes 5, 6, 7 et 8 sont accompagnées d'un dialogue pour guider l'utilisateur. Pour ce faire, un rectangle de 230 pixels en x et 80 pixels en y (en unités normalisées) centré dans la clôture active est utilisé. À la fin de l'exécution d'une commande de cette fonction, l'ancienne fenêtre est placée à la tête de la

liste circulaire, la nouvelle fenêtre est activée avec la fonction def-fenêtre et la clôture courante est effacée pour permettre de dessiner le contenu de la nouvelle fenêtre. Il est explicite que l'unité graphique sur laquelle on utilise cette fonction permette l'effaçage sélectif car cette fonction utilise cette caractéristique sans le vérifier.

Les fenêtres sauvées sur le disque sont écrites dans le fichier WINDOW. Il se trouve dans le répertoire graphique et il est créé s'il n'existe pas (voir section 2.1).

3.4.2 Primitives graphiques de haut niveau

Les fonctions regroupées ici permettent la création de primitives graphiques de haut niveau tant dans l'univers de l'usager que dans le système de coordonnées normalisées. Pour ces fonctions, le découpage ("clipping") est automatique.

Nous utilisons le paramètre booléen syst-coor pour indiquer à BGC le système de coordonnées à utiliser pour les paires XY.

Valeur de syst-coor	Constante (c-graph.h)	Signification
1	FEN	Coordonnées de la fenêtre
0	CLOT	Coordonnées de la clôture (normalisées)

Les paramètres utilisés pour décrire la taille des objets appartiennent à l'échelle des attributs. Ceux-ci doivent être invariants lors des changements de fenêtre. Pour ce faire, ils seront toujours interprétés en unités normalisées de l'écran. Nous rappelons que le motif et la couleur de tracé des vecteurs qui composent ces primitives graphiques sont ceux définis par les fonctions fixe-motif et fixe-couleur (voir section 3.3.2).

NOM

dess-arc

SYNOPSIS

```
void dess-arc (xc,yc,xd,yd,syst-coor,longueur,nb-seg)
float xc,yc,xd,yd ;     /*Appartiennent au système de coordonnées*/
float longueur    ;     /*de l'usager ou de la clôture selon     */
                        /*la valeur du paramètre syst-coor       */
short nb-seg      ;
boolean syst-coor ;
```

DESCRIPTION

Cette fonction dessine un arc de cercle centré au point de coordonnée (xc,yc) débutant à l'adresse (xd,yd). La longueur de l'arc est spécifié en radians et le nombre de segments qui le composent mesure la résolution voulue.

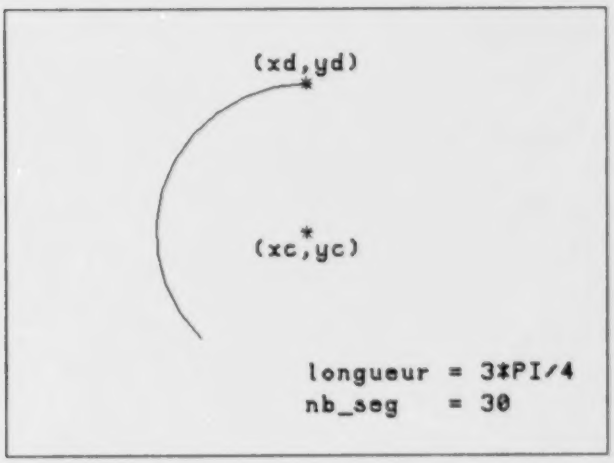

Figure 3.4 : Arc de cercle

NOM

dess-aster

SYNOPSIS

```
void dess-aster (xc,yc,syst-coor,rayon)
float xc,yc       ;
float rayon       ;
boolean syst-coor ;
```

DESCRIPTION

Cette fonction dessine un astérisque (étoile) centrée au point dont la coordonnée est (xc,yc). Le paramètre rayon appartient à l'échelle des attributs. Il est exprimé en unités normalisées.

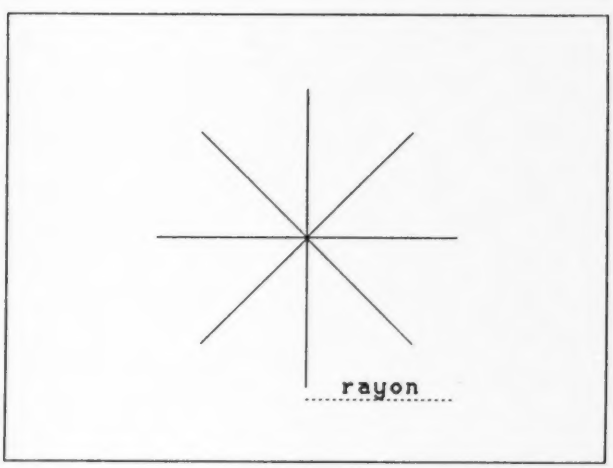

Figure 3.5 : Astérisque ou étoile

NOM

dess-cercle

SYNOPSIS

```
void dess-cercle (xc,yc,syst-coor,rayon,nb-seg)
float xc,yc       ;
float rayon       ;
short ng-seg      ;
boolean syst-coor ;
```

DESCRIPTION

Pour dessiner un cercle dont le centre se trouve au point de coordonnées (xc,yc). D'une manière semblable à la fonction dess-arc, le nombre de segments définit la résolution du cercle. Cette fonction peut aussi être utilisée pour tracer un polygone à nb-seg côtés.

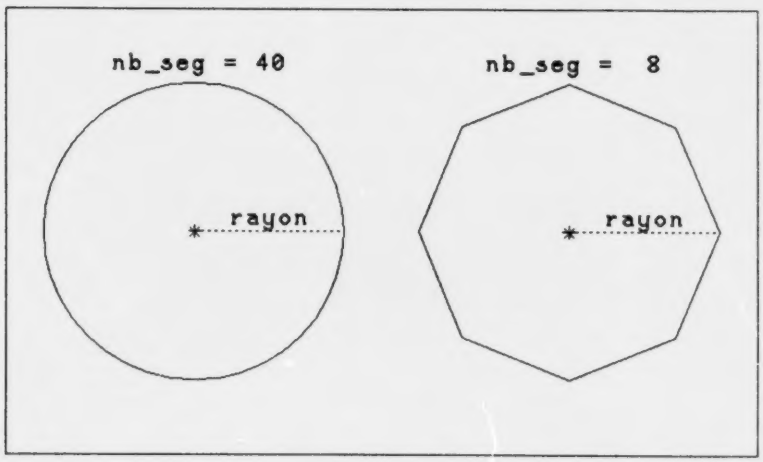

Figure 3.6 : Un cercle et un polygone

NOM

dess-échelle

SYNOPSIS

```
void dess-échelle (x,y,e)
float x,y,e ;
```

DESCRIPTION

Cette fonction dessine une échelle dont le coin inférieur gauche est situé à l'adresse (x,y) du système de coordonnées de la clôture. Le paramètre 'e' indique le nombre d'unités par pixel de l'écran normalisé. C'est la valeur de l'échelle.

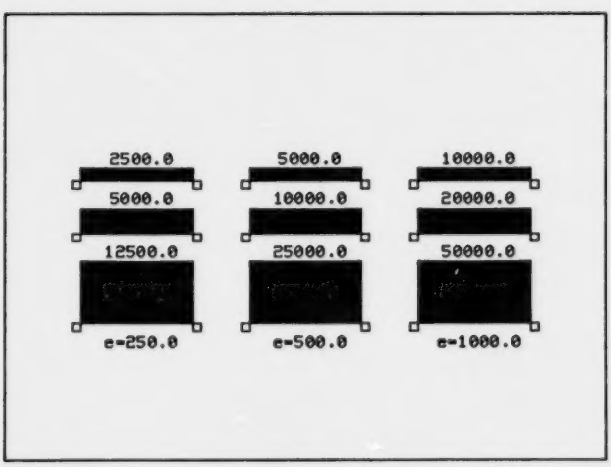

Figure 3.7 : Trois échelles

NOM

dess-flèche

SYNOPSIS

```
void dess-flèche (xd,yd,xf,yf,syst-coor,b,h)
float xd,yd,xf,yf ;
float b,y         ;
boolean syst-coor ;
```

DESCRIPTION

Cette fonction dessine un vecteur qui relie les deux points dont les coordonnées sont (xd,yd) et (xf,yf) dont l'extrémité est terminé par une flèche.

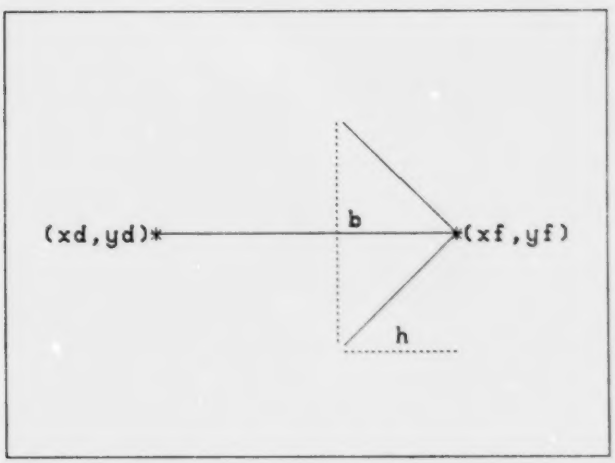

Figure 3.8 : Une flèche

NOM

dess-lien

SYNOPSIS

```
void dess-lien (x1,y1,x2,y2,t1,t2,d,e,v,c)
float x1,y1,x2,y2,d,e,v ;
int   t1,t2,c           ;
```

DESCRIPTION

Cette fonction dessine un lien entre deux sommets.

x1,y1 : coordonnée du centre du premier sommet;

x2,y2 : coordonnée du centre du second sommet (en unités de la fenêtre);

t1 : type du premier sommet;

t2 : type du second sommet

type	description
1	rectangle de taille 1
2	rectangle de taille 2
3	rectangle de taille 3
4	rectangle de taille 4
5	rectangle de taille 5
-1	cercle de taille 1
-2	cercle de taille 2
-3	cercle de taille 3
-4	cercle de taille 4
-5	cercle de taille 5

d : déplacement de la barre (nombre positif)
Note : la barre sera TOUJOURS dessinée à la GAUCHE du vecteur en suivant le sens du lien;

e : échelle = nombre d'unités par pixel de l'écran normalisé;

v : valeur numérique pour laquelle on veut une représentation graphique;

c : code de dessin
0 : dessiner un vecteur qui relie les deux sommets;
1 : dessiner un vecteur qui relie les deux sommets terminer par une flèche suivant le sens du lien;
2 : dessiner une barre pleine seulement représentant la valeur numérique;
3 : dessiner une barre vide seulement représentant la valeur numérique;
4 : dessiner la chaîne de caractères associée à la valeur numérique seulement sur une flèche;
5 : dessiner la barre (pleine) et la chaîne de caractères;
6 : dessiner la barre (vide) et la chaîne de caractères;

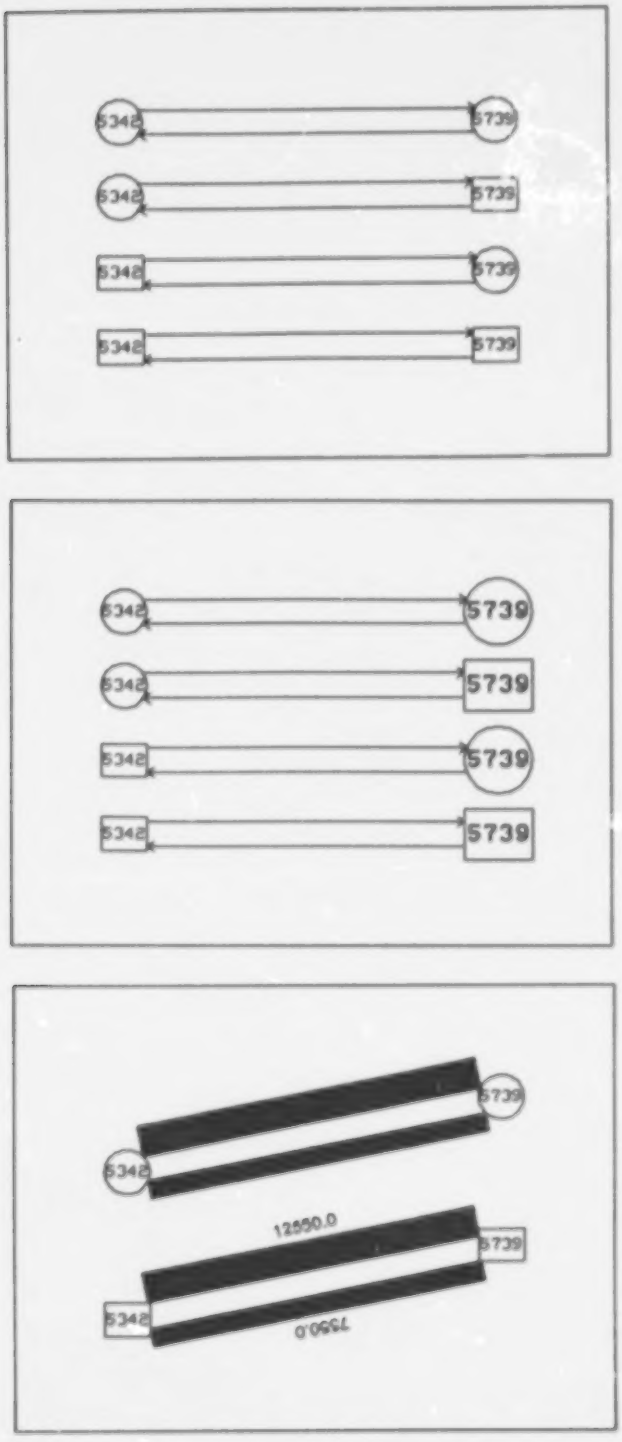

Figure 3.9 : Exemples de lien

NOM

dess-noeud

SYNOPSIS

```
void dess-noeud (x,y,t,mn,nu,c)
float x,y ;
int   t,c ;
char  *mn ;
```

DESCRIPTION

Dessiner une boîte autour d'un noeud et y inscrire le numéro du noeud ou son mnémonique.

x,y : position du noeud dans le plan cartésien (Coor. de la fenêtre);

t :

type	description
1	rectangle de taille 1
2	rectangle de taille 2
3	rectangle de taille 3
4	rectangle de taille 4
5	rectangle de taille 5
-1	cercle de taille 1
-2	cercle de taille 2
-3	cercle de taille 3
-4	cercle de taille 4
-5	cercle de taille 5

Taille	1	2	3	4	5
base	60.0	54.0	36.0	33.00	08.5
hauteur	45.0	40.5	27.0	24.75	08.5
rayon	30.0	27.5	18.0	17.00	08.5

mn : mnémonique (chaîne d'au plus quatre caractères terminée par NUL);

nu : numéro du noeud (4 chiffres au maximum);

c : code de remplissage du rectangle ou du cercle
- 0 rien
- 1 numéro
- 2 mnémonique
- 3 numéro inverse vidéo (pas implanté).

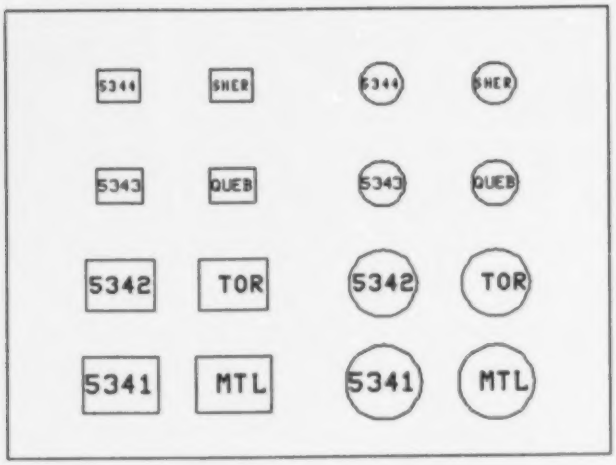

Figure 3.10 : Noeuds de différentes tailles

NOM

dess-rec

SYNOPSIS

```
int dess-rec (xc,yc,syst-coor,b,h)
float xc,yc       ;
float b,h         ;
boolean syst-coor ;
```

DESCRIPTION

Pour dessiner un rectangle. Les valeurs de retour de la fonction sont :

1 rectangle à l'intérieur du système de coordonnées selon la valeur du paramètre syst-coor.

0 rectangle partiellement à l'intérieur du système de coordonnées utilisé.

-1 rectangle comportement à l'extérieur.

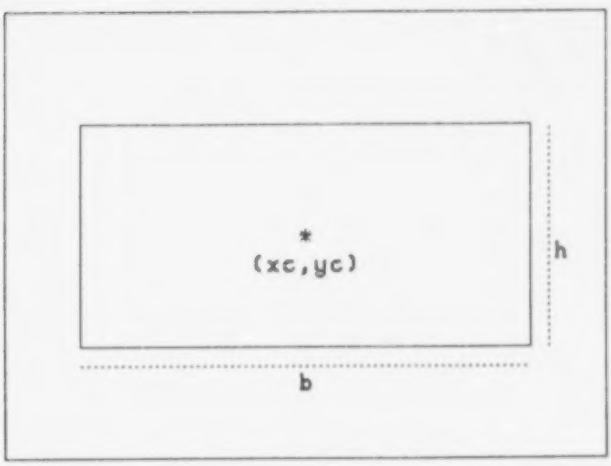

Figure 3.11 : Un rectangle

NOM

dess-vec

SYNOPSIS

```
int dess-vec (x1,y1,x2,y2,syst-coor)
float x1,y1,x2,y2 ;
boolean syst-coor ;
```

DESCRIPTION

C'est la fonction de base pour créer des primitives graphiques de haut niveau. Un vecteur dont le motif et la couleur ont été fixés avec les fonctions fixe-motif et fixe-couleur est dessiné. Il relie les points de coordonnées (x1,y1) et (x2,y2). De plus, cette fonction retourne les codes de découpage suivant :

0 La droite passe à l'intérieur de la clôture mais ses extrémités sont à l'extérieur.

1 La coordonnée (x1,y1) est à l'intérieur du système de coordonnées utilisé.

2 Le point de coordonnée (x2,y2) est à l'intérieur du système de coordonnées utilisé.

3 Le vecteur est complètement visible à l'intérieur de l'écran normalisé.

-1 Le vecteur est entièrement à l'extérieur de l'écran normalisé.

NOM

joint-cercle

SYNOPSIS

```
void joint-cercle (x1,y1,x2,y2,syst-coor,r1,r2,d)
float   x1,y1,x2,y2 ;
float   r1,r2,d     ;
boolean syst-coor   ;
```

DESCRIPTION

Cette fonction dessine un vecteur qui relie les centres de deux cercles. Les parties du vecteur qui sont à l'intérieur des cercles sont cachées. De plus, la droite peut êetre déplacée d'un certain nombre d'unités de l'écran normalisé vers la droite ($d > 0$) ou vers la gauche ($d < 0$) en suivant la direction du vecteur. Le sens du vecteur est obtenu en parcourant ce dernier du point 1 au point 2. Soulignons que les cercles ne sont pas tracés par cette fonction.

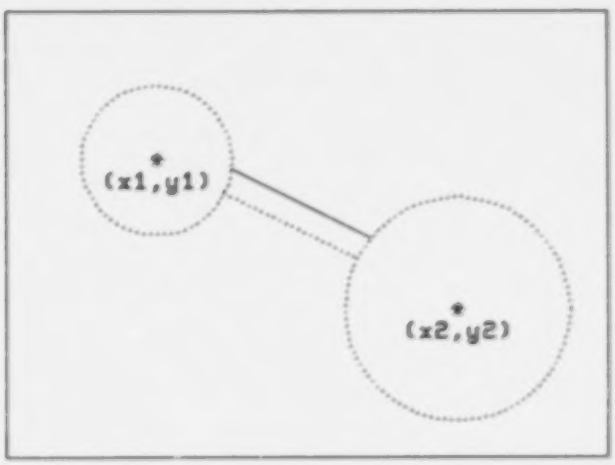

Figure 3.12 : Un vecteur qui relie deux cercles

NOM

joint-rec

SYNOPSIS

```
void joint-rec (x1,y1,x2,y2,syst-coor,b1,h1,b2,h2,d)
float x1,y1,x2,y2   ;
float b1,h1,b2,h2,d ;
boolean syst-coor   ;
```

DESCRIPTION

Cette fonction dessine un vecteur qui relie les centres de deux rectangles. Les parties du vecteur qui sont à l'intérieur des rectangles sont cachées. De plus, la droite peut être déplacée d'un certain nombre d'unités de l'écran normalisé vers la droite (d > 0) ou vers la gauche (d < 0) en suivant la direction du vecteur. Le sens du vecteur est obtenu en parcourant ce dernier du point 1 au point 2. Soulignons que les rectangles ne sont pas tracés par cette fonction.

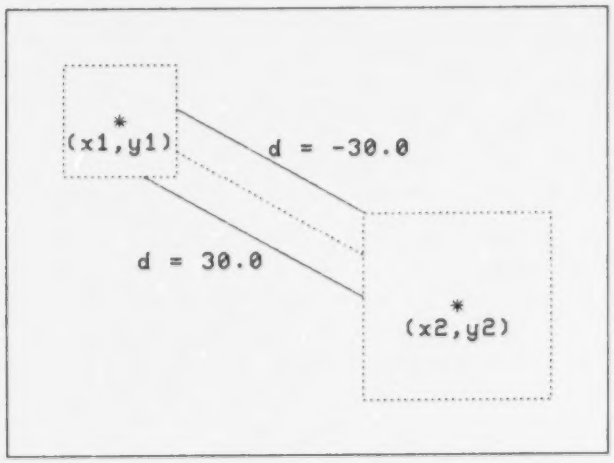

Figure 3.13 : Un vecteur qui relie deux rectangles

3.4.3 Générateur de caractères

NOM

dess-ch

SYNOPSIS

```
void dess-ch (x1,y1,x2,y2,syst-coor,chaîne,nb-car)
float x1,y1,x2,y2 ;
boolean syst-coor ;
short nb-car      ;
char *chaîne      ;
```

DESCRIPTION

C'est la fonction de base pour dessiner une chaîne de caractères dans une direction arbitraire. La fonte utilisée est celle définie par la fonction fixe-fonte. Par défaut, cette fonction utilise la fonte ASCII. La taille des caractères est fixée avec la fonction fixe-gross tandis que le motif et la couleur de tracé des vecteurs ont été activés avec les fonctions fixe-motif et fixe-couleur.

La coordonnée (x1,y1) indique le point de départ pour dessiner la chaîne de caractères tandis que la coordonnée (x2,y2) est un point sur le vecteur directeur qui soutient la chaîne. Soulignons que le vecteur directeur n'est pas dessiné.

NOM

dess-ch-vec,dess-nb-vec

SYNOPSIS

```
void dess-ch-vec (x1,y1,x2,y2,syst-coor,d,chaîne,nb-car)
float x1,y1,x2,y2 ;
float d           ;
boolean syst-coor ;
short nb-car      ;
char *chaîne      ;
void dess-nb-vec (x1,y1,x2,y2,syst-coor,d,chaîne,nombre,format)
float x1,y1,x2,y2 ;
float d           ;
float nombre      ;
boolean syst-coor ;
char *format      ;
```

DESCRIPTION

dess-ch-vec dessine une chaîne de caractères sur le vecteur invisible d'origine (x1,y1) et de terminaison (x2,y2). La chaîne est centrée sur ce vecteur et est déplacée d'un certain nombre d'unités de l'écran normalisé.

dess-nb-vec permet de dessiner un nombre sur un vecteur invisible d'origine (x1,y1) et de terminaison (x2,y2). Le nombre est transformé dans sa chaîne de caractères correspondante selon le format spécifié.

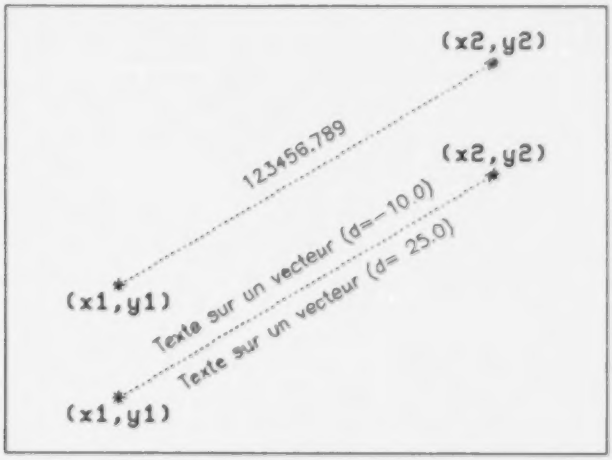

Figure 3.14 : Chaîne de caractères et un nombre sur un vecteur.

3.4.4 Opérations mathématiques

Les fonctions présentées ici effectuent des opérations mathématiques sur des vecteurs. Elles ne produisent pas de dessin sur la surface de visualisation.

NOM

dep-vec

SYNOPSIS

```
void dep-vec (x1,y1,x2,y2,x1d,y1d,x2d,y2d,syst-coor,d)
float x1,y1,x2,y2 ;          /*vecteur initial                    */
float *x1d,*y1d,*x2d,*y2d ; /*Résultat du déplacement           */
                             /*exprimé en unités du système      */
                             /*de coordonnées spécifié par       */
                             /*syst-coor                         */
float d                     ; /*d > 0 déplacement vers la droite*/
                             /*d < 0 déplacement vers la gauche*/
```

DESCRIPTION

Pour déplacer un vecteur d'un certain nombre d'unités normalisées de l'écran.

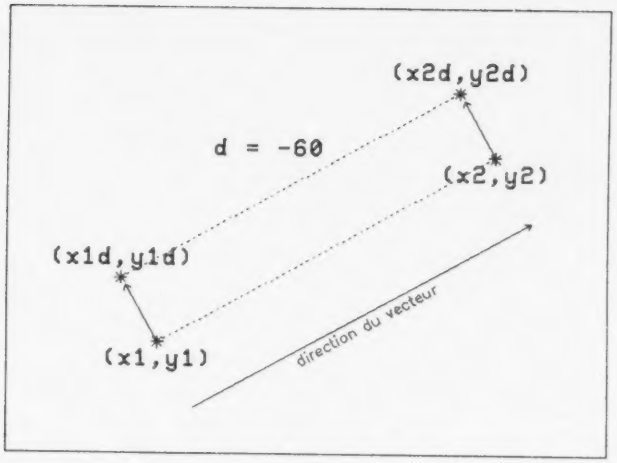

Figure 3.15 : Déplacement d'un vecteur

NOM

equat-droite

SYNOPSIS

```
int equat-droite (x1,y1,x2,y2,pente,béta)
float x1,y1,x2,y2,pente,béta
```

DESCRIPTION

Cette fonction calcule les paramètres de l'équation d'une droite étant donné deux points de celle-ci. On se rappelle qu'on peut définir une droite par l'équation :

$$y = pente * x + bêta$$

où $pente = \frac{x1 - x2}{y1 - y2}$ la pente est infinie si le dénominateur est nul. Dans ce cas, c'est une droite verticale.

bêta = l'ordonné à l'origine

De plus, la fonction retourne les deux valeurs suivantes :

1 si la pente est infinie
Dans ce cas, le paramètre de sortie pente prend la valeur de la constante INFINI (voir c-glob.h).

0 sinon

NOM

inter-vec

SYNOPSIS

```
int inter-vec (xa1,ya1,xa2,ya2,xb1,yb1,xb2,yb2,xi,yi)
float xa1,ya1,xa2,ya2 ; /*Premier vecteur     */
float xb1,yb1,xb2,yb2 ; /*Second vecteur      */
float *xi,*yi         ; /*Point d'intersection*/
```

DESCRIPTION

Pour calculer l'intersection entre deux vecteurs.

Valeurs de retour :

-1	vecteurs parallèles
0	vecteurs confondus
1	l'intersection a été trouvée

Lorsque le point d'intersection n'a pu être calculé, la coordonnée (xi,yi) reste indéterminée.

NOM

inter-cer-vec

SYNOPSIS

```
int inter-cer-vec (x1,y1,x2,y2,r,d,xi,yi)
float x1,y1,x2,y2 ;
float r           ; /*rayon du cercle*/
float d           ;
float *xi,*yi     ;
```

DESCRIPTION

Pour calculer l'intersection entre un vecteur et un cercle.

Valeurs de retour :
- 1 si l'intersection est trouvée
- 0 sinon, dans ce cas la coordonnée (xi,yi) reste indéterminée

Soulignons que tous les paramètres sont exprimés avec des unités du même système de coordonnées.

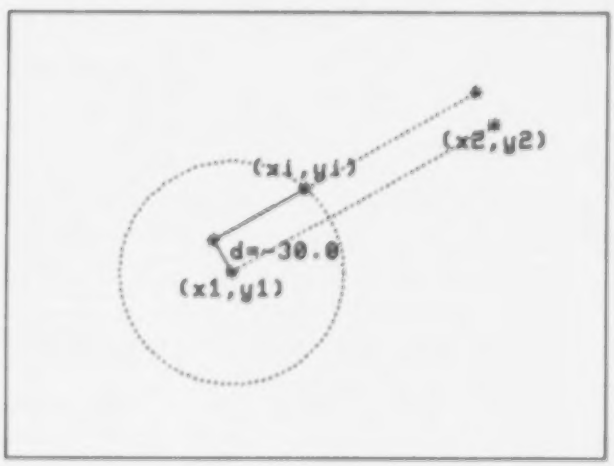

Figure 3.16 : Intersection d'un vecteur avec un cercle

NOM

inter-rec-vec

SYNOPSIS

```
int inter-rec-vec (x1,y1,x2,y2,xc,yc,b,h,xi,yi)
float x1,y1,x2,y2 ;     /*vecteur                        */
float xc,yc       ;     /*centre du rectangle            */
float b,h         ;     /*base et hauteur du rectangle*/
float *xi,*yi     ;     /*point d'intersection           */
```

DESCRIPTION

Pour calculer l'intersection entre un vecteur et un rectangle.

Valeurs de retour :

1 si l'intersection est trouvée

0 sinon, dans ce cas la coordonnée (xi,yi) reste indéterminée

Soulignons que tous les paramètres sont exprimés avec des unités du même système de coordonnées.

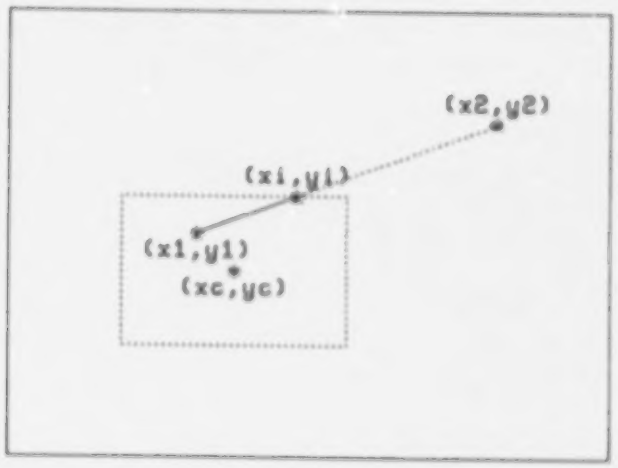

Figure 3.17 : Intersection d'un vecteur avec un rectangle

CONCLUSION

Nous espérons que le travail que nous venons de présenter contribuera à justifier le vieux dicton chinois qui dit qu'un dessin vaut mille mots.

Nous avons exposé les concepts de base pour la mise en oeuvre d'une bibliothèque graphique. Nous avons vu comment la structure de BGC respecte les standard graphiques en restant indépendante des ordinateurs et des périphériques graphiques. Ensuite nous avons décrit l'utilisation de ses fonctions à l'intérieur d'un programme d'application.

Nous admettons que cette première version de BGC constitue une bibliothèque graphique simple. Par contre, elle est bien adaptée au graphisme de réseau et elle forme un noyau informatique puissant pour la construction des systèmes interactifs graphiques appliquées à la recherche opérationnelle.

Le choix du langage a été judicieux, car il a permis d'écrire des fonctions efficaces et transportables. Nous soulignons que BGC a été développé sur un ordinateur SUN utilisant un système d'exploitation UNIX. Nous avons aussi implanté BGC sur un micro-ordinateur IBM XT. Cette implantation a nécessité une modification mineure dans un fichier en-tête pour tenir compte du système d'exploitation DOS.

Nous nous sommes efforcés d'employer un style de programmation clair et modulaire pour faciliter les améliorations futures. Entre autres, il sera intéressant d'implanter l'utilisation des segments et de permettre le graphisme 3D. De plus, nous prévoyons le développement d'interfaces de bas niveau avec d'autres standards graphiques tel ceux employés par IBM et de générer du code POST SCRIPT pour les imprimantes au laser.

BIBLIOGRAPHIE

CRAINIC, T.G., DEJAX, P. (1987), "The Use of Computer Interactive-Graphic Systems in the Modelling of Distribution and Transportation Systems", INFORS '87, Buenos Aires.

CRAINIC, T.G., DEJAX, P., DELORME, L. (1986), "Problèmes de localisation-distribution avec équilibrage : éléments de modélisation et propriétés", publication #490F, Centre de recherche sur les transports, Université de Montréal.

CRAINIC, T.G., MONDOU, J.-F. (1986), "Systèmes interactifs graphiques pour la planification des réseaux de transport interurbain des marchandises - Méthodologie et premiers résultats", publication #467, Centre de recherche sur les transports, Université de Montréal.

FISHER, M.L. (1985/1986), "Interactive Optimization", Annals of Operations Research 5, 541-556.

FLORIAN, M., CRAINIC, T.G., GUÉLAT, J., BABIN, A. (1987), "STAN: An Interactive-Graphic System for Strategic Transportation Analysis at the National and Regional Levels", publication #495, Centre de recherche sur les transports, Université de Montréal.

FOLEY, J.D., VAN DAM, A. (1984), "Fundamentals of Interactive Computer Graphics", Addison Wesley.

HOROWITZ, A.J., PITHAVADIAN, A.R. (1987), "Generalized Computer Aided Design of Transportation Networks", Transportation Quarterly 41(3), 397-409.

KERNIGHAM, B.W., RITCHIE, D.M. (1983), "Le langage C", Masson, Paris.

PEARSON, R. (1987), "Aspects avancés du langage C", Interédition, Paris.

SPIESS, H. (1984), "Contribution à la théorie et aux outils de planification des réseaux de transport urbain", publication #382, Centre de recherche sur les transports, Université de Montréal.

WOLCOTT, N.M., HILSENRATH, J., "A Contribution to Computer Typesetting Techniques: Tables of Coordinates for Hershey's Repertory of Accidental Typefonts and Graphic Symbols".

ANNEXE A

ascii.h	Définition des caractères ASCII non imprimables
c-glob.h	Constantes globales
c-graph.h	Constantes graphiques
io-mac.h	Macro d'entrée-sortie
matmac.h	Macro commandes à caractère mathématique
termcap.h	Unités graphiques soutenues par BGC

```
/* ************************************************************************** */
/*                                                                            */
/*                      +-----------------------------------+                 */
/*                      |                                   |                 */
/*                      | ascii.h                           |                 */
/*                      |                                   |                 */
/*                      | auteurs  : Sylvie Couture         |                 */
/*                      |            Jean-Francois Mondou   |                 */
/*                      |                                   |                 */
/*                      +-----------------------------------+                 */
/*                                                                            */
/* Definition des caracteres ASCII non imprimables .                          */
/*                                                                            */
/* ************************************************************************** */

#define   NUL           '\000'
#define   SOH           '\001'
#define   STX           '\002'
#define   ETX           '\003'
#define   EOT           '\004'
#define   ENQ           '\005'
#define   ACK           '\006'
#define   BELL          '\007'
#define   BACK_SPACE    '\010'
#define   H_TAB         '\011'
#define   LINE_FEED     '\012'
#define   V_TAB         '\013'
#define   FORM_FEED     '\014'
#define   RETURN        '\015'
#define   SHIFT_OUT     '\016'
#define   SHIFT_IN      '\017'
#define   DLE           '\020'
#define   DC1           '\021'
#define   DC2           '\022'
#define   DC3           '\023'
#define   DC4           '\024'
#define   NACK          '\025'
#define   SYN           '\026'
#define   ETB           '\027'
#define   CANCEL        '\030'
#define   EM            '\031'
#define   SUB           '\032'
#define   ESCAPE        '\033'
#define   FS            '\034'
#define   GS            '\035'
#define   RS            '\036'
#define   US            '\037'
#define   SPACE         '\040'
```

```
/* ************************************************************************** */
/*                                                                            */
/*                     +--------------------------------+                     */
/*                     |                                |                     */
/*                     | c_glob.h                       |                     */
/*                     |                                |                     */
/*                     | auteurs  : Sylvie Couture      |                     */
/*                     |            Jean-Francois Mondou|                     */
/*                     |                                |                     */
/*                     +--------------------------------+                     */
/*                                                                            */
/* Definition des constantes globales                                         */
/*                                                                            */
/* ************************************************************************** */

/* Constantes generales */
#define     TRUE            1
#define     FALSE           0
#define     SENTINELLE      -1
#define     IO_ERROR        -1
#define     ERROR           1
#define     NO_ERROR        0

/* Constantes a caracteres mathematiques */
#define     PI              3.141592653
#define     DEUX_PI         6.283185306
#define     DEMI_PI         1.570796326
#define     EPSILON         0.001
#define     INFINI          1000000.0
#define     UNDEFINED       -9999.9999
#define     MAXCAR_LIGNE    132

/* Definition des types */
typedef  short  boolean;
typedef  char   buffer [BUFFER_SIZE + 1] ;
```

```
/* ************************************************************************** */
/*                                                                            */
/*                      +-------------------------------+                     */
/*                      |                               |                     */
/*                      | c_graph.h                     |                     */
/*                      |                               |                     */
/*                      | auteurs  : Sylvie Couture     |                     */
/*                      |            Jean-Francois Mondou|                    */
/*                      |                               |                     */
/*                      +-------------------------------+                     */
/*                                                                            */
/* Constantes graphiques                                                      */
/*                                                                            */
/* ************************************************************************** */

/* Constantes utilisees pour l'effacage selectif */
#define  SOLID_FILL    '@'
#define  GREY_FILL     'A'
#define  DOTS_ON       '0'
#define  DOTS_OFF      '1'
#define  COMPLEMENT    '2'

/* Motif de trace des vecteurs */
#define  INVISIBLE      0
#define  PLEIN          1
#define  POINT          2
#define  PT_TRAIT       3
#define  PE_TRAIT       4
#define  LG_TRAIT       5

/* Couleur de trace de vecteurs */
#define    NOIR         0
#define    BLANC        1
#define    ROUGE        2
#define    VERT         3
#define    BLEU         4
#define    CYAN         5
#define    MAGENTA      6
#define    JAUNE        7
#define    ORANGE       8
#define    VERT_J       9
#define    VERT_C      10
#define    BLEU_C      11
#define    BLEU_M      12
#define    ROUGE_M     13
#define    GRIS_F      14
#define    GRIS_P      15

/* Commandes de la fonction window */
#define     CURSEUR     1
#define     CENTRE      2
#define     ZOOM        3
#define     COORD       4
#define     DROITE      5
#define     GAUCHE      6
#define     HAUT        7
#define     BAS         8
#define     RAPPELER    9
#define     DERNIERE   10
#define     SAUVER     11

/* Constantes de systeme de coordonnees */
#define    FEN          1
#define    CLOT         0
```

```
#define     BUFFER_SIZE    512
#define     STRING_SIZE     32

/* Fontes pour le generateur de caracteres */
#define     ASCII            0
#define     FONTE_1          1
#define     FONTE_2          2
#define     LIBERE          -1
```

```
/* ************************************************************************** */
/*                                                                            */
/*                      +--------------------------------+                    */
/*                      |                                |                    */
/*                      | io_mac.h                       |                    */
/*                      |                                |                    */
/*                      | auteurs  : Sylvie Couture      |                    */
/*                      |            Jean-Francois Mondou|                    */
/*                      |                                |                    */
/*                      +--------------------------------+                    */
/*                                                                            */
/*  Macros d'entree sortie                                                    */
/*                                                                            */
/* ************************************************************************** */

#define is_control (x) ((x > 0 && x < 32) ? 1 : 0)
```

```
/* ************************************************************************** */
/*                                                                            */
/*                     +--------------------------------+                     */
/*                     |                                |                     */
/*                     | matmac.h                       |                     */
/*                     |                                |                     */
/*                     | auteurs  : Sylvie Couture      |                     */
/*                     |            Jean-Francois Mondou|                     */
/*                     |                                |                     */
/*                     +--------------------------------+                     */
/*                                                                            */
/* MACRO commande a caractere mathematique .                                  */
/*                                                                            */
/* Reference  : The C library                                                 */
/*              -------------                                                 */
/*              Kris Jamsa                                                    */
/*                                                                            */
/*      1) _MAX (x, y)                                                        */
/*      2) _MIN (x, y)                                                        */
/*      3) _ABS_VAL (x)                                                       */
/*      4) _ROUND_OFF (x)                                                     */
/*      5) _SIGNE (x)                                                         */
/*      6) _SGN_BIT (x)                                                       */
/*                                                                            */
/* ************************************************************************** */

#define _MAX(A, B)          (((A) > (B)) ? (A) : (B))
#define _MIN(A, B)          (((A) < (B)) ? (A) : (B))
#define _ABS_VAL(A)         (((A) >= 0) ? (A) : (0 - (A)))
#define _ROUND_OFF(A)       (int ((A) + 0.5))
#define _SIGNE(A)           ((A) > 0 ? 1 : -1)
#define _SGN_BIT(A)         ((A) >= 0 ? 0 : 1)
```

```
/* ************************************************************************
/*
/*                      +--------------------------------+
/*                      |                                |
/*                      | termcap.h                      |
/*                      |                                |
/*                      | auteurs  : Sylvie Couture      |
/*                      |            Jean-Francois Mondou|
/*                      |                                |
/*                      +--------------------------------+
/*
/*
/* Unites graphiques soutenues par BGC.
/*
/* ************************************************************************

/* Constantes assignes aux unites graphiques */
#define     NB_UNITES    6
#define     TK4014       1
#define     VIS500       2
#define     VIS550       3
#define     TK4111       4
#define     TK4115       5
#define     TK4663       6

/* Noms assignes (chaines de caracteres) aux unites graphiques */
unite        TERM_NAME [NB_UNITES] = { "TK4014",
                                       "VIS500",
                                       "VIS550",
                                       "TK4111",
                                       "TK4115",
                                       "TK4663"  } ;
```

ANNEXE B

L'utilitaire mcd ("modify current device") permet à l'usager de sélectionner d'une manière fiable l'unité graphique courante pour chacune des deux classes.

C'est un programme interactif où l'usager est guidé par des menus. Il y a deux niveaux de menus. Le premier indique à l'utilisateur l'état actuel du fichier cur-dev et lui permet de charger l'unité graphique courante de la classe TERMINAL ou de la classe PLOTTER. Le second niveau imprime à l'écran la liste des appareils graphiques soutenus par BGC. C'est à ce stade que l'usager spécifie la nouvelle unité graphique courante. Lorsque la modification est complétée, l'utilisateur est ramené au premier niveau où il peut demander une autre modification ou quitter le programme.

```
**************************
**                      **
**        (mcd)         **
** Modify Current Device **
**                      **
**************************
```

```
Current device for classe TERMINAL is TK4014
Current device for classe PLOTTER  is TK4014


1) Modify current device for classe TERMINAL
2) Modify current device for classe PLOTTER
3) Quit
-> 1
```

```
** Modify current device for classe TERMINAL **


Devices are :    0) TK4014
                 1) VIS500
                 2) VIS550
                 3) TK4111
                 4) TK4115
                 5) TK4663

Enter new current device -> 3
```

```
**********************************
**                              **
**          (mcd)               **
** Modify Current Device        **
**                              **
**********************************
```

```
Current device for classe TERMINAL is TK4111
Current device for classe PLOTTER  is TK4014


1) Modify current device for classe TERMINAL
2) Modify current device for classe PLOTTER
3) Quit
-> 2
```

```
** Modify current device for classe PLOTTER  **


Devices are :   0) TK4014
                1) VIS500
                2) VIS550
                3) TK4111
                4) TK4115
                5) TK4663

Enter new current device -> 5
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx                          xx
xx          (mcd)           xx
xx Modify Current Device    xx
xx                          xx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
Current device for classe TERMINAL is TK4111
Current device for classe PLOTTER  is TK4663


1) Modify current device for classe TERMINAL
2) Modify current device for classe PLOTTER
3) Quit
-> 3
```